Tracking system for Bitcoin colouring

Major author: Llorenç Escoter Torres

Supervisors: Lisardo Prieto González and Ricardo Colomo Palacios

I bow before the authority of special men because it is imposed on me by my own reason.

Mikhail Bakunin

Acknowledgments

To everyone who is responsible for making this document possible.

tumak_, thanks for the light you gave me on the colour aware block chain.

FellowTraveller, the work you do in Open Transactions looks amazing.

Abstract

Nowadays Bitcoin is the main reference for virtual currency. Because it is decentralized and with an open implementation, there are several online currencies that follow the same structure.

Analysing Bitcoin definition one realises of the possibility of colouring them. There exist many methods of implementing such colouring but, after several months of discussion, the community decided to implement what is known as Ordering Bitcoin Colouring (OBC).

Colouring systems require for the creation of what is known as colour definition. These are files which describe the name of the colour, the means of identifying it; they can include several signatures and certifications.

With the existence of such colours, appears the need for exchanging the definitions and trading the coloured coins. Currently, the only way of doing so is by means of a central server. Such centrality does not belong to what is desired in decentralized networks. This is why it is appealing to build a system that can circumvent the central server.

This project provides a definition of the structure of such system, as well as a working prototype as a proof of concept.

Keywords
Bitcoin, coloured Bitcoin, distributed definition system, distributed trading system, CB2CB

Table of Contents

Index of figures

1 Index of tables

1 Introduction

1.1 Motivation

BitcoinX[1] provides colour support for the Bitcoin Network by using order based colouring[2]. However, colour definitions are currently stored in a central server, and the exchanges are posted on an online board on the same server. This introduces a single point of failure. Coloured Bitcoins can no longer be issued or traded if the sever is taken down. From a completely decentralized point of view this scenario is not desirable at all. One of the main goals of Bitcoin is such decentralization, and this is not a feature users are willing to give up. It is possible that users decide not to use coloured coins because of how vulnerable the current system is.

This project provides a general structure of how the system could be rebuilt in order to remove the single point of failure. Both colour defining and trading will be taken into account, empowering the user to perform those actions in a decentralized fashion.

1.2 Document organization

In order to ease the reading of the present document, in this section we detail how it has been divided into several chapters and information those contain.

Chapter 1, Introduction, provides us with an introductory view of the proposed system. It also includes the motivations and objectives to be satisfied by the project.

Chapter 2, State of the art, contains an analysis of the current situation of Bitcoin and its current colouring system. Several solutions such as Bitcoin and BitcoinX are described and analysed. Information contained in this chapter will be useful for the analysis and design of the proposed system.

Chapter 3, Description of the Solution, contains the analysis, design, implementation, and testing of the software. All the information provided here is useful to understand what the system does and how it does it, as well as alternative solutions.

Chapter 4, Examples of usage, provides the reader with some scenarios where the software's usefulness can be seen. It can also be considered as an illustration of the system at work.

Chapter 5, Conclusions and future lines of work, contains the conclusions extracted from the development of the project. It can also be used as a guide to expanding the software.

Annex 1, Glossary, contains brief definitions on many of the concepts used in the document.

Annex 2, Project management, details on the planning of the project, the budget planned and deviations from it.

Annex 3, User manual, provides users with a brief manual on how to use the added functionality.

1.3 Goals

Coin colouring has many uses, basically we move from a one-dimensional coin to a multi-dimensional one. Of the many possible implementations, it has been decided to implement what is known as an "order based colouring".

The goal of the project is to provide P2P distribution of the colour definitions, as well as creating a simple P2P market that allows you to offer your coloured Bitcoin coins (from now on, BTC) in exchange for BTC.

1.3.1 P2P distribution of the colour definitions

Colour definitions are files that store JSON text and, in a near future, should include signatures, proper descriptions, certifications, and other fields that will make the files grow larger. Currently, their size is relatively small, less than 1 kilobyte. This is because nowadays they include very limited information; provided information relates to the name and initial transaction of the colour. However, if coloured coinage becomes a standard, definitions' size is expected to grow largely due to inclusion of new fields, such as long descriptions or digital certificates.

It is intended to provide the means for users to share these definitions amongst themselves. It must be possible to do so in a decentralized manner and with the least network overhead.

1.3.2 P2P market

Knowing that users can share their created coins' definitions, it seems desirable to enable users to trade them. This trading can be done in several ways, for example, there can be several trading places, where users join to make their offers. This usage is already possible so it will no longer be discussed. However, we can create a P2P trading network where users can post their offers and find peers who want to match them.

Several components have been considered that are useful in this case.

1.3.2.1 Offer relaying

Peers will be relaying other peers' offers to ensure that the network can find a match for them. The policy followed for offer relaying will be simple; all offers are to be relayed unless they previously have been so.

1.3.2.2 Offer matching

Peers might match two offers; in such event, they would have to tell each peer how to contact the other one.

1.3.2.3 Token trading

In a multi-dimensional coin system, atomic transactions are really useful. If we carefully create the transaction, we can make exchanges atomic.

Because transactions can have several inputs and outputs, and multiple signatures can be required to make a transaction valid, we can construct a single transaction with the inputs of all the parties, create the outputs so that all parties can agree, and then make all the parties sign it.

2 State of the art

The classical approach of creating a currency relies on a trusted entity, such as a Central Bank, to define the properties of the currency it creates. Typically a Central Bank would be printing bills or tokens and, maybe, claiming that those tokens are backed by gold. It is quite easy to realise that mentioned gold can be replaced by **trust on the issuer**. Money can be accepted as a part of an exchange, not because is backed by gold but because it is *scarce*. Also, money is accepted as long as you know that others will accept it too. This allows for the Central Bank to get rid of its gold, as long as it keeps the currency in a condition where many people will accept it.

However, with the previous approach, we need to either be a highly trusted entity or have some good to back our currency. We also need to know that the authority issuing the currency has control of how the money is produced; this gives the authority the potential to do many bad things such as issuing more money than promised.

Bitcoin is capable of removing that central authority from the money printing process. This is done by defining how the currency will work a-priori and not allowing any changes unless most of the hashing power agrees on them.

2.1 Bitcoin

Bitcoin is P2P distributed crypto-currency[3]. It uses the current cryptographic mechanisms[4] to provide software that behaves like coinage. Note that, despite the fact that the coin is based on cryptographic principles, no information is cyphered at any point. The mechanisms used include hashing and asymmetric cyphering to sign transactions[5].

2.1.1 Benefits

Bitcoin solves the problem of centralized authority in the production of digital currency. Up until its appearance, all digital currency systems have been based on some kind of central entity. Bitcoins allows the **creation of coinage based on scarcity instead of a central authority**, creating a **virtual currency that anyone can produce**, much like the Rai stones[6].

Users can freely trade BTCs around the world, **without any entity being able to control who owns the coinage[7]**.

The creation of a new account requires that the user computes an Elliptic Curve (from now on, EC) key pair. Note that this process is almost **free**; it just costs the electricity of the computation.

2.1.1.1 Anonymity

If properly used, Bitcoin Network can protect user's identity and become **completely anonymous**. Even though the connections are neither ciphered nor hidden, one can *torify* the connections to provide the much desired anonymity[8]. Accounts are identifiable; however, users can create as many accounts as desired, by means of EC key pairs, and, by this, obtain account anonymity.

2.1.1.2 Decentralized

There is no central part in the Bitcoin Network. No central authority is required to issue or trade the money. This removes the power from central entities such as the *American Federal Reserve*. This also takes the power from governments devaluate peoples currency, as long as they use Bitcoin.

2.1.2 How it works

Bitcoin bases all of its strength in the *proof of work* idea. Transactions are grouped into blocks that need to be solved. Once a *miner* solves a block he can spend the BTC reward as well as the fees of all the transactions he included in the block.

Around the world there is a lot of what is known as *hashing power*. This computing power is used to try to solve Bitcoin blocks in order to claim the reward. The more *hashing power* that is introduced the harder the blocks get to solve, making Bitcoin Network more secure. This difficulty update takes place every 2016 blocks, approximately two weeks[9].

As can be seen in Figure 1, Bob can send BTCs to Alice and Alice can check her account. To allow this to happen, *miners* keep collecting transactions and adding them to blocks. Once they find a solution to the block, they can keep the transactions' fees as well as the block reward, 25 BTC at the present time[10].

Bob will add a small fee to his transaction, for the *miner* to keep, as an incentive for *miners* to include it in the next block they produce.

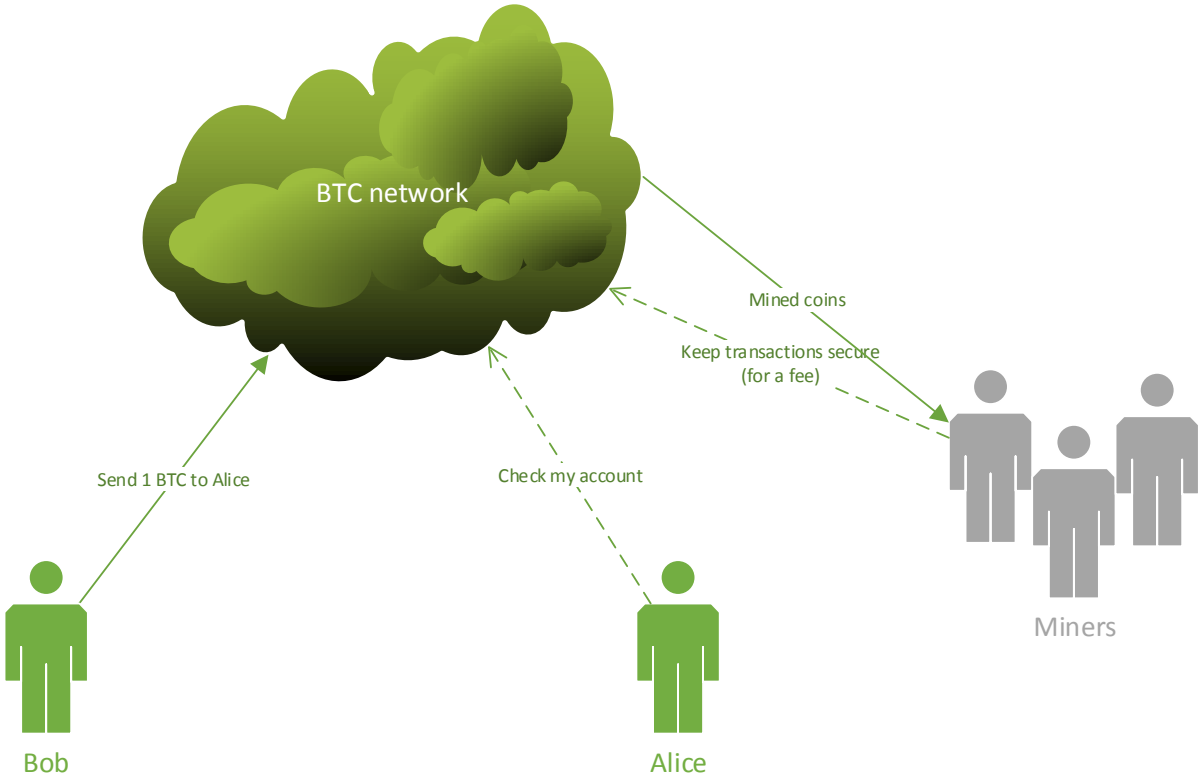

Figure 1 Bob sending money to Alice

2.1.2.1 Scarcity

Scarcity is a need in all monetary systems. It makes almost no sense to have a currency which is not scarce, where everyone has as much as they want. For example, scarcity in € is controlled by the interest rate of European Central Bank (ECB from now on), and the requirements in accessing such credits[11].

The scarcity on the Bitcoin Network, on the other hand, is based on the *proof of work* concept. If you want to be able to spend some coins, you must prove that you have worked enough to get those coins, or that you obtained them from someone else. This proof is just a **computationally hard to solve, yet easy to verify, problem**[12]. The problem used can have its difficulty adjusted to solve a block each 10 minutes. The solver of the block then obtains a reward in the form of BTC, currently 25, plus the fees that were introduced in the transactions[13].

2.1.2.2 Transactions

Every time we send BTCs to an account we generate what is called a **transaction**. Transactions behave in an **atomic way**, either all the transaction is valid or none of it.

2.1.2.2.1 Definition

A transaction consists of a series of inputs, a set of outputs, and, usually, a fee. Each output also includes a verification script, written as a **stack automaton**[14]. Transactions can only be collected by those who can provide an input to the verification script that makes the automaton end in a true state. Providing this input is what is called collecting a transaction (see Figure 3).

2.1.2.2.2 Properties

Transactions are malleable enough for us to produce all kinds of special transactions. This includes transactions that require multiple signatures, transactions that can be spent by anyone, and so on. We can produce any transaction that we can encode as a stack automaton with the provided language[15]. However, the automaton language is not at its full potential, much of the functionality is disabled[16]. As long as we only use accepted code for our stack automaton, transactions created will be valid.

2.1.2.2.3 Structure

Field	Description	Size (bytes)
Version no	Currently 1	4
Input counter	Variable length integer[17]	1 – 9
List of inputs	List of inputs of the transaction	Variable
Output counter	Variable length integer	1 – 9
List of outputs	List of outputs of the transaction	Variable
Lock time	Timestamp describing when the transaction will be valid. It can either specify blocks or time.	4

Table 1 Transaction structure

Field	Description	Size (bytes)
Previous Transaction Hash	Double SHA256 of a previous transaction to be used	32 bytes
Previous output index	Non negative integer indexing an output of the to-be-used transaction	4 bytes
Input script length	Non negative variable length integer	1 – 9
Input script	Script	Input script length
Sequence no	Usually 0xFFFFFFFF; irrelevant unless transaction's lock time is greater than 0	4 bytes

Table 2 Input transaction structure

Field	Description	Size (bytes)
Value	Non negative integer with the number of Satoshis[18]	32 bytes
Output script length	Non negative variable length integer	1 – 9
Output script	Script	Output script length

Table 3 Output transaction structure

To verify a transaction, input scripts are combined with their respective output scripts. First we put the input script, and then we append the output one. After this, we execute the automaton and the collection is valid if the automaton ends in a true state.

```
Output script: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

Input script (collection): <sig> <pubKey>
```

Figure 2 Standard transaction to a Bitcoin address

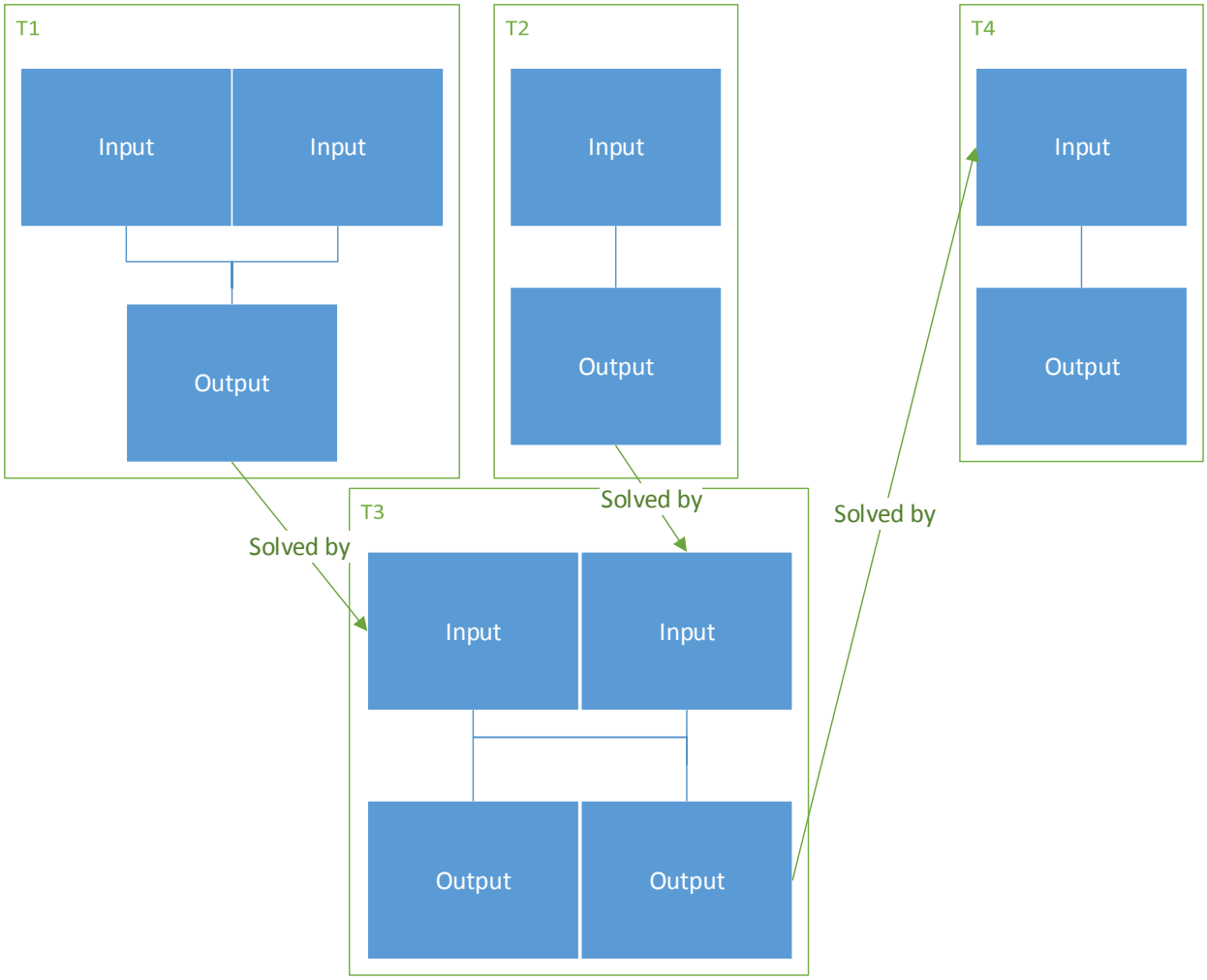

Figure 3 Chained transactions

2.1.2.3 Blocks

Transactions are grouped into *blocks*. These build the knowledge of the Bitcoin Network regarding who owns what. Blocks' difficulty is adjusted so that, on average, one block is generated every ten minutes[19].

2.1.2.3.1 Definition

Blocks are files, stored at each node of the network, where data is permanently recorded. A block contains a record of transactions not included in previous blocks. We can also a find a reference to the previous block, which allows us to generate what is known as a block chain and a solution to the block. They can be seen as pages of a ledger where all transactions since the previous page are stored.

2.1.2.3.2 Properties

A solved block is one that verifies the following formula:

$$SHA256\big(SHA256(block\ header)\big) \leq Bits$$

Figure 4 Block solving equation

Transactions are appended after the block's header. A *Merkle tree*[20] (see Merkle tree) is built upon them and its root is stored in the header. Because transactions are not hashed, but their *Merkle root* is, it takes the same amount of computing power to solve a block with 1 transaction as it takes to solve one with 10,000 transactions. Difficulty is adjusted every 2016 blocks to ensure that an average of 6 blocks per hour have are produced[21].

Blocks can have several solutions. In fact, multiple block solvers (aka *miners*) can solve blocks with different transactions. In the exceptional case where two *miners* solve a block at, approximately, the same time it is possible to find a fork in the chain. This is known as a *blockchain* split but the Bitcoin Network is conveniently designed to be resilient to it. Clients only consider valid the longest chain, measuring length as the combined difficulty of the chained blocks. **This prevents anyone from creating a long chain of easy blocks**.

2.1.2.3.3 Structure

Table 4 shows the structure of a block.

Field	Description	Size (bytes)
Magic no.	Constant 0xD9B4BEF9, other coins such as Litecoin[22] use another one.	4
Blocksize	Number of bytes following up to end of block	4
Block header	Header of the block, this is what is used to solve it	80
Transaction counter	Variable length integer	1-9
transactions	The list of transactions	variable

Table 4 Block structure

Of all the information above, only the block header is used by the block solving mechanism. This allows increasing the size of blocks without having to increase the amount of data of the problem. The header includes a merkle tree hash root of the transactions.

Field	Purpose	Updated when…	Size (bytes)
Version no	Block version number	You upgrade the software and it specifies a new version	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all the transactions in the block	A transaction is accepted	32
Time	Current timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current target in compact format	The difficulty is adjusted	4
Nonce	32-bit number	A hash is tried	4

Table 5 Block header structure

2.1.2.3.4 Merkle tree

Merkle trees are extremely useful for large data verification. As it can be seen below, *Merkle trees* can be used to hold lots of transactions. They also provide a *single point proof* for the rest of the data[23].

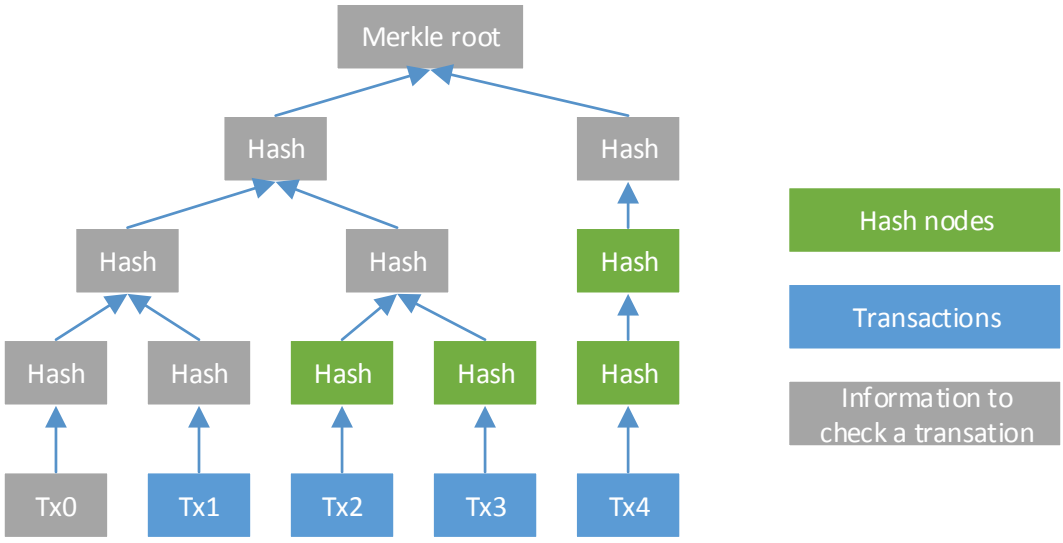

Figure 5 Merkle hash tree

The construction of such trees depends on the order of the transactions, and because of the way transactions are distributed[24], so having two *miners* working on the same *Merkle root* is almost impossible. This means that there is no need for communication between *miners* to avoid trying the same solutions because each *miner* has a different problem.

2.1.2.4 Block Chain

The *block chain* represents the public and secure ledger of the entire Bitcoin Network. This is the revolutionary idea required to give birth to Bitcoin. In fact, this structure can be used to *provide DNS, P2P currency exchanges, SSL certificate authorities, time stamping, file storage, and voting systems* all in a **decentralized manner**, without the need of central authorities[25].

The structure can also be used to create many systems that usually require trust, without the need for a central authority. All these structures can be integrated in the same securing network; As Satoshi said, networks based in this kind of structure can share computing power[26]. Hopefully, this will allow current ASICs such as Avalon's[27] to be used to secure other networks as well.

2.1.2.4.1 Definition

The *block chain* is the database containing all the transactions since the creation of the Bitcoin Network. Using this database, users can keep track of all the transactions and know how many BTCs each account has. The chain can be built because every block contains a reference, **in the means of a hash**, to the previous block[28].

2.1.2.4.2 Properties

Each block can only be computed, solved, once the previous one has. This occurs because the hash of the previous block cannot be known before solving it. This fact **guarantees that blocks are generated chronologically**. It also makes computationally impractical to modify previously created blocks because it would imply modifying all the following blocks in the chain.

Honest *miners* will only try to solve a block referencing the last block of the longest chain. Chains are only valid if all the blocks are valid, link to the *genesis block*, and are the *longest one*. Length is measured as the combined difficulty of all the blocks in the chain, thus preventing *dishonest miners* to erase previous transactions.

2.1.2.4.3 Structure

Because all blocks reference, by means of a hash, the previous block, from every block there is only one path to the genesis one. The opposite, however, is not true; if we start from the genesis block we can find some **forks**.

Forks containing a single block are created when two *miners* solve a block simultaneously. When this occurs it is up to the *miners* to decide what block to mine on. Once another block is solved the tie is broken, and we get a single block fork.

Longer forks can take place after every software change that is not backward compatible. We can imagine a scenario where there is a group of *miners* using software that is not compatible with the rest of them. This will create a big fork where each of the groups will have its own block chain. This can be a big trouble because block chains cannot be merged.

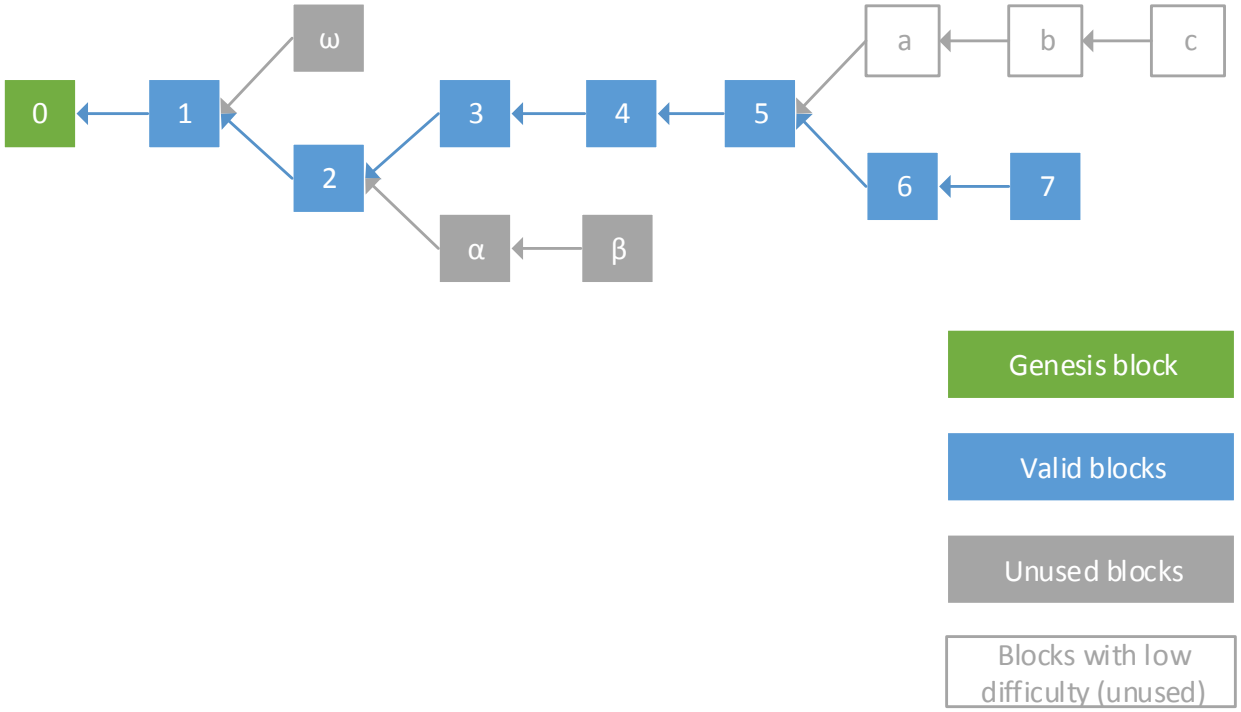

Figure 6 Block chain structure

In Figure 6 we can see an example of a block chain. We can easily notice the examples of all that was mentioned previously.

We can find a one block fork in (ω, 2) that could have been generated by two *miners* solving a block at the same time. The fork gets solved when 3 or α appear and the *miners* stop working on the block following ω.

With the blocks α and β we can see how a fork provoked by a non-backward compatible update, where a group continued working with a version and did not accept the new blocks, might look like. This is also known as a **hard fork**.

Finally, with blocks a, b, and c, we can see *how a fork provoked by a malicious user might look like*. However, we must keep in mind that the fork would be disregarded due the low difficulty of the blocks.

2.1.3 Issues and risks

Having mentioned all the wonders of Bitcoin, it is now time to move on to its issues. Minor issues such as those caused by *lowering of Bitcoin prices*, *lack of computer security*, and *lack of backups* will not be mentioned here.

2.1.3.1 51% attack

A *miner* holding more than 50% of the hashing power is *virtually capable* to erase any transaction he desires. Because he has more hashing power than the rest of the network, he can make his own chain and, eventually, have built a chain longer than the legitimate one. Even if this attack might seem remote and improvable **it is still an event to take into consideration**[29].

2.1.3.2 Legal issues

Many countries simply do not consider the possibility of coinage without control. It is not uncommon for *virtual currency* legislations to rely on central authorities issuers of their own coinage and legislations tend to require such central authority[30]. No Spanish legislation on decentralized currencies, such as Bitcoin, has been found during the research period of this document. Also, depending on the products one issues with the coloured coins, legal concerns might arise.

2.1.3.3 Quantum computing

Bitcoin transactions make use of ECDSA[31] algorithm. Elliptic curve cryptographic is sensible to *Shor's algorithm*[32] and, with the appearance of *quantum computing* anyone might be able to spend any transaction[33]. This implies that, whenever quantum computers make their appearance, it is probable that all that is based in asymmetric cyphering stops being secure.

2.1.3.4 Block chain forks

Block chain forks are **unavoidable**. Probably, in less than 12 months, another hard block chain fork is to be required. Transaction volume has been doubling every 4 months[34] and, because the current limit size of blocks is 1 MB, blocks will no longer be able to hold all the required transactions.

Even if they are unavoidable, there is a major problem related to block chain forks, the uncertainty that imposes in the users. Their transactions might no longer be valid if they are no longer included in the longest chain.

2.2 BitcoinX

BitcoinX[35] is a software project forked from **Armory**[36] which is capable of Bitcoin colouring and, with minor modifications, it also can colour all Bitcoin based currencies, such as *Litecoin*. It does so by working on top of an existing block chain structure, maintained by the original client, taking advantage of the current security of the network and without having to modify any of the existing clients.

2.2.1 Coloured Bitcoins

Coloured Bitcoins are ordinary Bitcoins that a set of users agrees they have another meaning. This can include, but is not limited to, the issuing of mining bonds, the trading of any cryptographic value[37], or the issuing of participations.

Before this can happen, users must agree on a couple of things. First, they must provide a way to allow the creation and definition of such colours. It must be a **univocal definition**, upon which all users must agree. Once a group agrees upon a definition method for the colours, they must also **propagate the value**. In other words, they need a way to transfer the coloured coins among themselves.

2.2.1.1 Benefits

Miners are not currently mining blocks because of the fees, but **because of the block reward**. By the time of writing the current block reward is of 25 BTCs[38], the default fee for the standard (Satoshi's) client is 0.0005 BTC/KB, and the maximum block size is 1 MB. This implies that the maximum expected revenue from fees is $\frac{0.0005\ BTC\ 1\ MB}{KB} = 0.512\ BTC$, much lower than the reward of 25 BTCs.

A day will come when block reward is simply not profitable enough. When this day arrives it is assumed that mining will continue to be profitable because of the **fees of the network**. The expansion of the dimension of the coins, by means of the introduction of colours, can dramatically increase the amount of transactions to be processed by the network and, in turn, increase the fee revenue. As more transactions are introduced in a block, *miners* can collect larger sums of fees.

Coloured Bitcoins allow us to trade any kind of commodity with similar, if not the same, properties as those in the Bitcoin Network. The trading of such colours is as secure as the Bitcoin Network, while keeping the decentralized infrastructure. This permits us to have an ecosystem with the benefits of a banking system without such a system.

2.2.1.1.1 Extremely general

Coloured coinage is so general that can be used in many situations. Any use case involving a party generating its own coinage, such as someone writing a check, can be modelled by means of coloured coinage. For example, companies such as *Ticket Restaurant* are forced to issue their own currency. However, they face the problem of creating something very much like money without the tools money issuers usually have access to[39]. A lot of effort is put in preventing counterfeit in the € bills, and not so much effort can be put by every entity issuing their own currency.

With coloured coinage, any entity can create its own currency, with as much protection as Bitcoin, without having to develop any kind of infrastructure. We can consider it similar to signing € bills. A group of friends can agree on that 5€ bills signed by one of them will be worth 50€, and that the signee is the warranty for it. In such scenario, 5€ bills will have the security of € bills, plus the security of the signature.

2.2.1.1.2 Easy storage

Coloured Bitcoins can be stored using the same systems that allow for storing of Bitcoins. In particular, **by using multiple signature transactions, coloured coins can be stored safely**, even if some of the accounts get hacked[40].

Also, with the apparition of physical wallets, wallets that can be backed up with paper, value can be stored as securely as desired.

2.2.1.1.3 Digital transfer without central authorization

Nowadays, value can be transferred with no need of a central authorization. This is achieved by **using coins like Bitcoin**.

2.2.1.1.4 Atomic transactions

Bitcoin **transactions are atomic**, they cannot be split. This implies no counterparty risk at all with the transactions, and allows for coloured coins to be traded for other coloured, or uncoloured, coins. The malleability of the transactions permits complex trades, such as exchange between different block chains[41].

2.2.1.1.5 Anonymity

If the proper tools are used, such as VPNs and *TOR*, anonymity can be achieved, and still enjoy the benefits of ownership.

2.2.1.1.6 The same infrastructure can be used

If the colouring is implemented on top of the Bitcoin protocol there is no need for users to update their clients; colour transactions are a specific case of Bitcoin ones. Clients without support for coloured transactions can still relay and verify the transactions, the only difference being they would not realise that the transactions they sign involve coloured coins.

2.2.1.2 Colour definition

As explained before, there is a requirement for a way of generating colour definitions. It is reasonable to expect the definitions not to be counterfeited. For example, a definition could be described in the terms of:

"The reward of the next block mined by account X will be sent evenly to the holders of coins of this colour" Signed by account X, DNI signature.

Several ways to define a colour are described below.

2.2.1.2.1 Genesis coin creation

We could state that coins created at a given block have a specified colour. A description could be issued where the coin properties are given. If that description was signed by the *miner* of the coins there would be no doubt about what the coins colour was[42].

This way of generating coloured coins, however, imposes strong restrictions on who can create them. It is not practical that the only ones who can provide colours are the *miners*.

2.2.1.2.1.1 Structure

Field	Description
Block hash	The hash of the block to be coloured
Description	The description of the colour
Name	The name of the colour
Colour id	The hash of Block hash, Description, and name
Signature	The signature of colour id
Public key	The public key of the signature

Table 6 Mined colour structure

The signature and public key would prevent users from colouring coins they do not own.

2.2.1.2.2 Genesis transaction

Another possible way of specifying a colour is by providing a genesis transaction. An owner of some Bitcoins can create a transaction, and then provide a definition involving such transaction. One would

include a description, with the corresponding *signatures* and *warranties*, to gain the trust of the potential buyers of the coloured coins.

2.2.1.2.2.1 Structure

Field	Description
Output index	The index of the outputs that will be coloured
Transaction hash	The hash of the genesis transaction
Name	The name of the colour
Colour id	The hash of Transaction hash, Output index, and Name

Table 7 Transaction colour structure

This is the structure used by BitcoinX to define colours. Theoretically, one would include things like signatures, and warranties inside the Name field. It should be noted that it is possible to issue a colour definition of a transaction completely unrelated to you.

2.2.1.2.3 Genesis account

Another possible definition of Bitcoin colours is to define the colour by means of a genesis account. This account would act as a mint; all the transactions leaving the account would be painted by the colour defined. This approach could be as robust as the transaction based colouring.

2.2.1.2.3.1 Structure

Field	Description
Account	A public key, representing a Bitcoin account
Name	The name of the colour
Colour id	The hash of Account, and name
Signature	The signature of colour id, using Account as the public key.

Table 8 Mined colour structure

We must take into account that, if this structure were used, fees paid to the network must not be considered[43] as coloured.

2.2.1.3 Propagation rules

Once we know how to define Bitcoin colours, we need a way to propagate them among the users. The Bitcoin protocol imposes that inputs of the transactions must add up to the outputs plus the transaction fee.

It is quite easy to verify that, if nothing is done to correct it, it is impossible to trace the coloured coins. Imagine a transaction with four inputs and two outputs. Assume that one of the inputs is blue, another is green, the other is purple, and the last one is not coloured at all. They all sum *S*, then the transaction will be valid as long as both outputs plus the fee sum S. However, there is no way to trace outputs to given inputs. This can be seen in Figure 7.

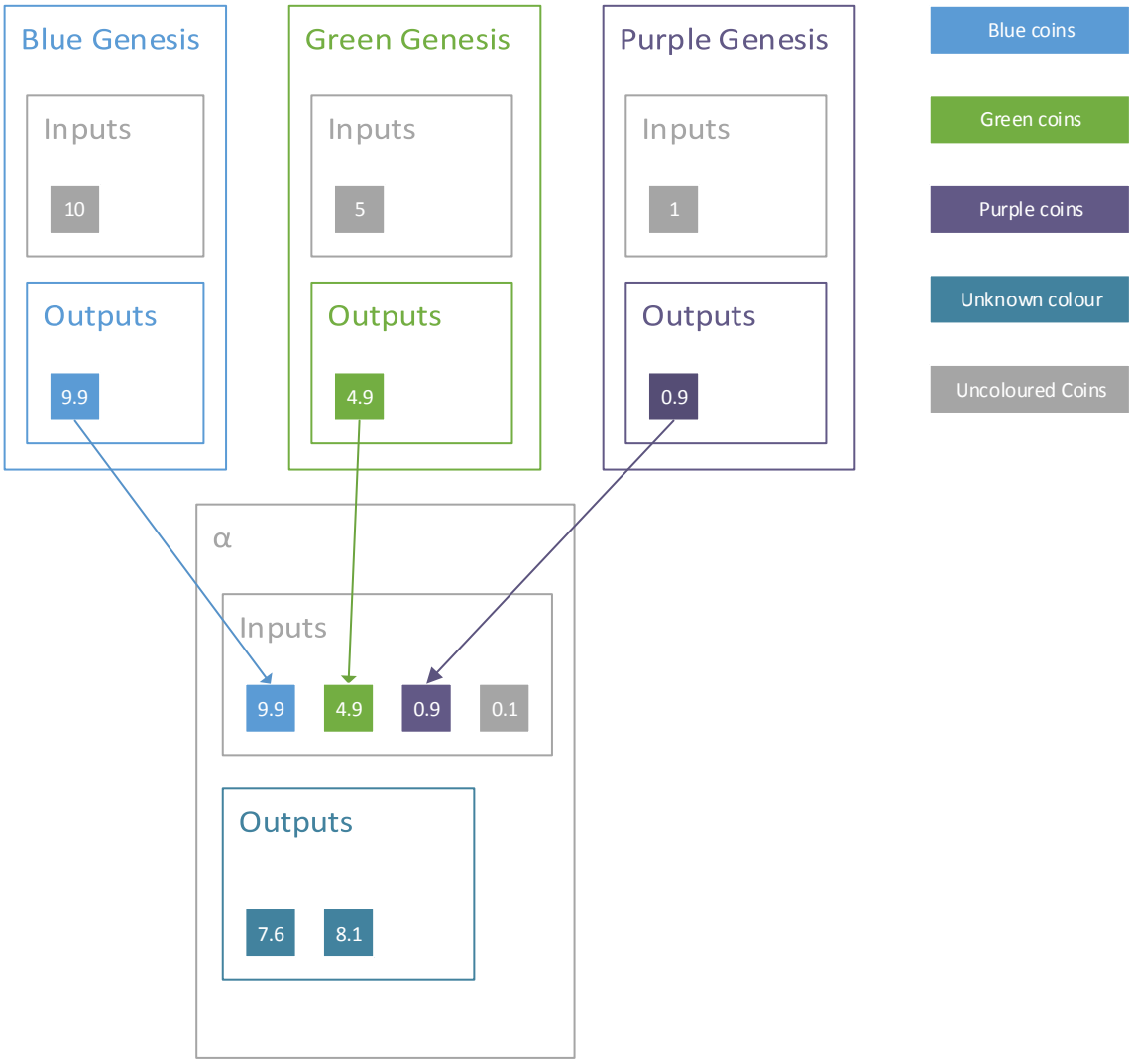

Figure 7 Untraceable colour transactions

As we can see in the previous example, it is impossible for us to know the colour of the outputs, unless we do something smart related to the transactions. We can try several solutions to keep track of the colouring of the coins.

2.2.1.3.1 Diluted colours

We could consider a mixing and **diluting of the colours**. If we had a transaction with red and blue inputs, we could mix and split the colouring among the outputs. It could be seen as mixing coloured waters; once they are mixed there is no way to split them again. Also, it could be considered that no colour is lost at all on fees, thus avoiding spending coloured coinage on fees.

This method is not really practical. Simple transactions such as Bob and Alice trading blue and red coins, respectively, are impossible in an atomic way. They must have to either agree to **do two separate transactions**, with the risks associated, or simply dilute their colours. If that were the case both Alice and Bob would end up with **coins that would be both partially blue and red**.

2.2.1.3.1.1 Example

Below is an example of how the previously described rules would apply.

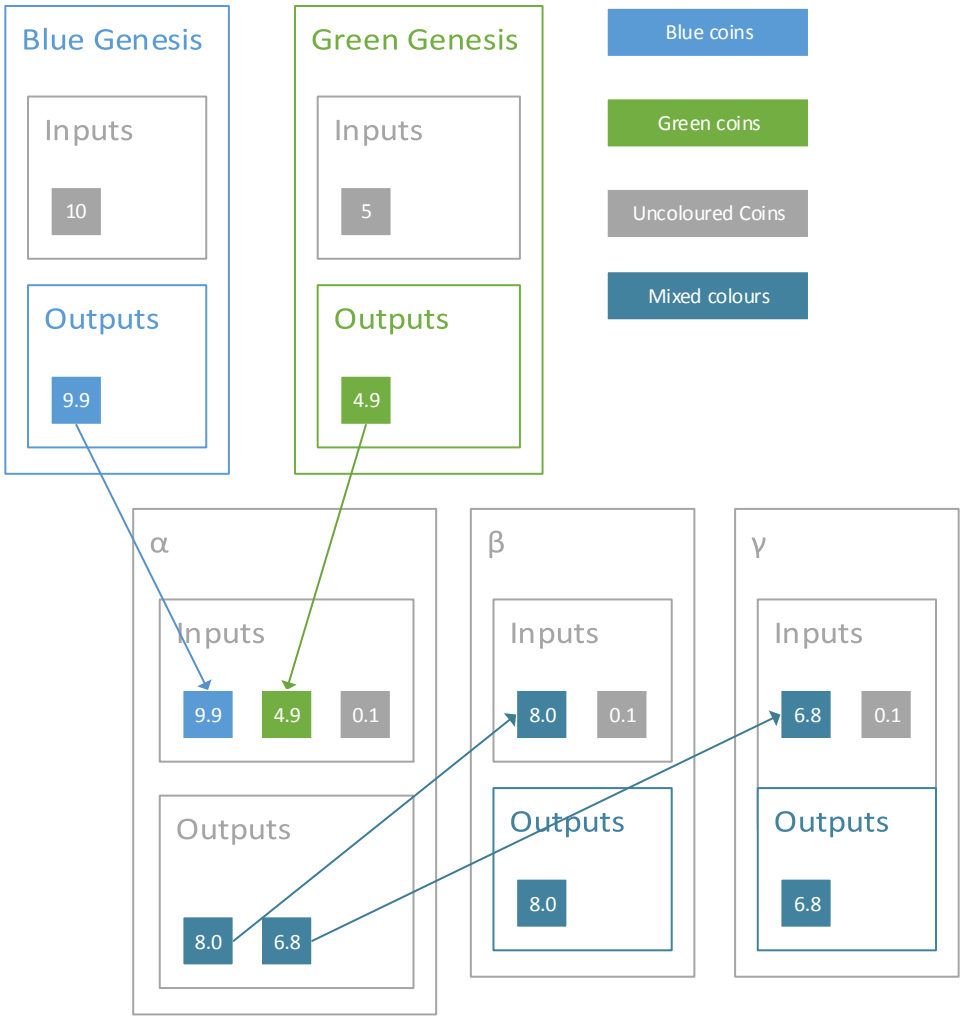

Figure 8 Diluted colour transaction

As we can see, outputs from transaction α are of a mixed colour from the inputs. We would say that they are of a mixed colour (66.89% blue and 33.11% green). Once the colours get mixed it is impossible to undo the mixing. That is something not so much desirable.

2.2.1.3.2 Monochrome transactions

Another possible solution to trade coloured Bitcoins could be to use **monochrome transactions**. If transaction's inputs are the entire same colour, but the one for the fee, then the colour of the output becomes clear.

This method still presents the problem of **not allowing atomic transactions** of multiple colours. This reason is sufficient to make it **virtually useless**.

2.2.1.3.2.1 Example

Below is an example of how the previously described rules would apply.

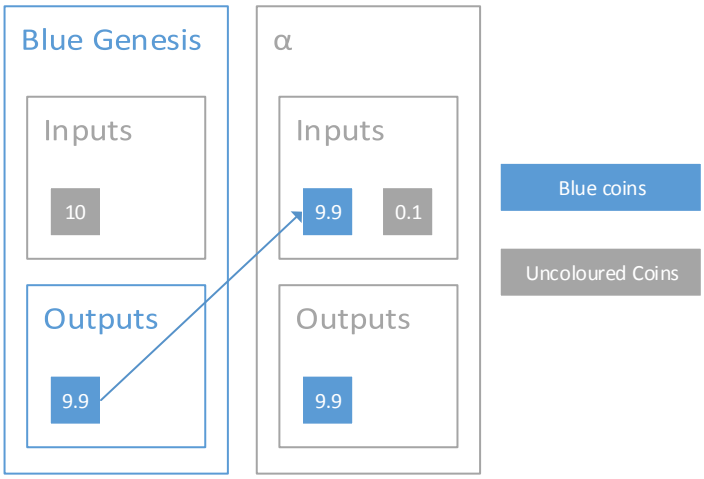

Figure 9 Monochrome transaction example

Monochrome transactions are stable in the sense that there is no colour mixing or *colour leakage* (see section Colour leaking) possible. However, if transactions do not include non-coloured inputs to pay for the fee, users might still end up paying fees with coloured coins.

2.2.1.3.3 Colour aware transactions and block chain

Another approach is to have a **colour aware protocol**. This would solve many of the current issues regarding coloured Bitcoins (see section Possible issues). If every input and output of each transaction contains a colour tag and is only relayed, and added to blocks, if the colours are conserved, we can have much faster algorithms for transferring coloured value. Currently, if using the Bitcoin Network, full scans are needed to detect a coin's colour; by the introduction of colour tags this would reduce to a constant time.

There is an initiative related to this approach known as alternative chain[44]. However, it is not still clear if Bitcoin will adopt colour tags for its transactions or users will eventually be forced to join the alternative block chain for their coloured coins. Because the problem with including colour tags in Bitcoin is a community's political one, it is not clear what the solution might be. It is possible that the two networks coexist as long as *merged mining*[45] becomes a more popular approach.

While Bitcoin does not include colour tags and **there is not enough hashing power** in in the alternative block chain **to make it secure**, we must keep on searching for methods of transferring coloured value in the current Bitcoin network. If all the hashing power from the Bitcoin Network moves to the alternate block chain they can produce enough *malicious blocks* to break it.

2.2.1.3.4 Order based colouring

Finally, the jewel in the crown; order based colouring[46] provides the means to generate atomic transactions of various colours. Also, these transactions do not require any modification of the clients. In

fact, all *miners* can currently relay these transactions for a relatively large fee due to the big size of generated transactions.

The algorithm is rather simple. We will have all the inputs and outputs of each transaction sorted by colour, and having the uncoloured ones at the end. By doing this it is possible to keep track of all the colours of the transaction, and have all the data in a single transaction.

2.2.1.3.4.1 Example

Below is an example of how the previously described rules would apply.

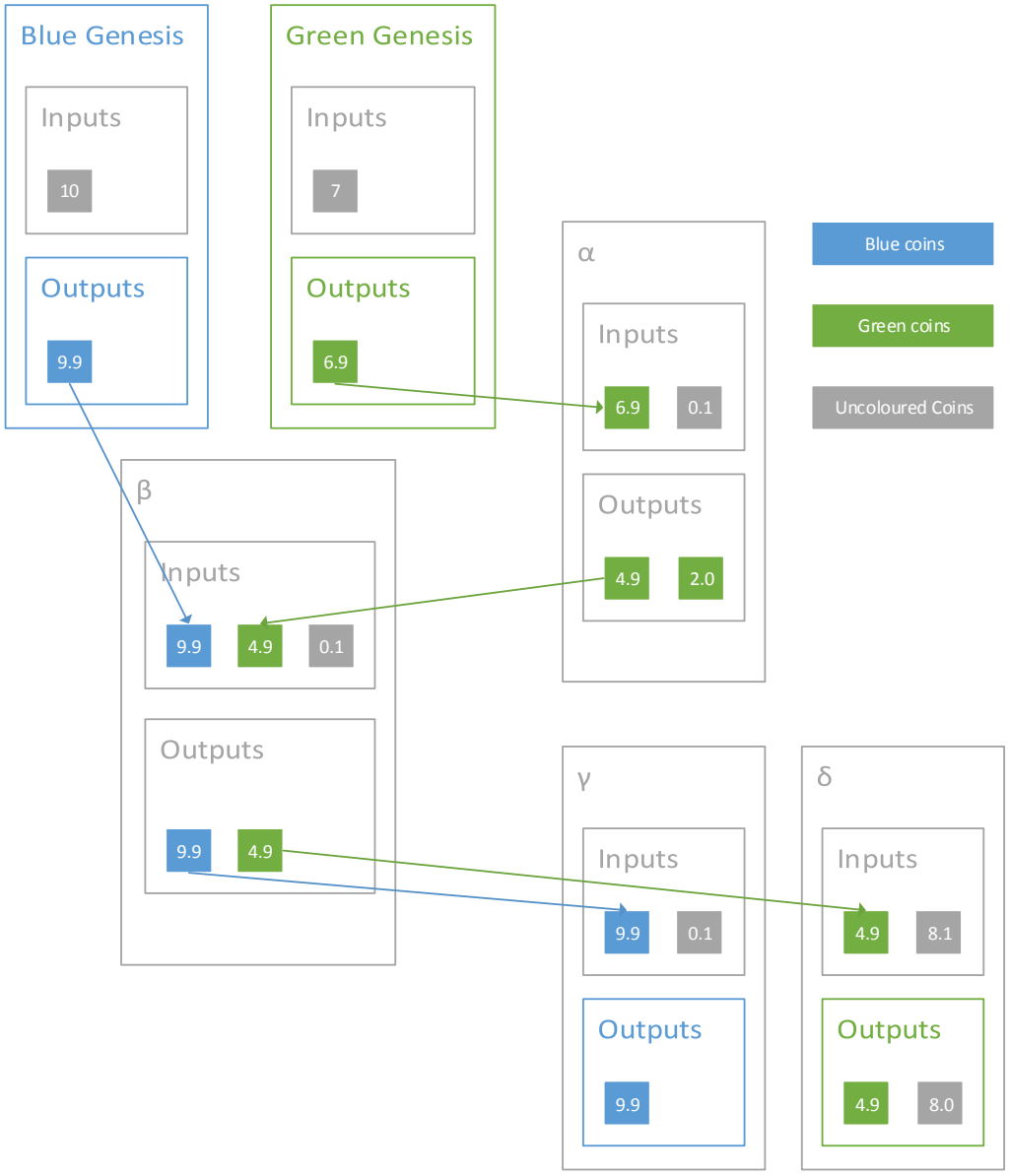

Figure 10 Order based colouring example

In the example above we can observe how users can keep track of the colours and trade both coloured and uncoloured coins in atomic transactions. For example, in transaction δ we can observe how a

trading of two colours might look. Also, in transaction β we can observe how to keep track of the colours as long as the inputs and the outputs are ordered by the *genesis transaction*.

2.2.1.4 Colour query specification

Given a certain transaction, we must **verify its colours**. This can be achieved in **two ways**. We can either locate the genesis transaction and follow the coins forward, or we can track the transaction backwards and check if it comes from any given genesis transaction.

2.2.1.4.1 Forward scanning

Assuming we have a transaction's output and we want to detect what colour it is, we can forward scan the block chain. This would be done by fetching each genesis transaction and following the chain of transactions, until the end of each line is reached or the transaction you had an interest in is found.

2.2.1.4.2 Backward scanning

When we have a transaction's output, and **assuming it has been built using order based colouring**, we can track it down to several inputs of the same colour. Then we should check the outputs that are collected by those inputs, and keep moving backwards until all of the transactions reach the genesis transaction. If all the transactions reach the genesis one, we know what the colour is. If not all the transactions reach the genesis, then there must have been some issues and no one should accept such transaction.

2.2.1.5 Possible issues

Because colour unaware clients exist, it is possible that a client that does not know what the colour of a certain transaction output is. If the client collects such transaction we cannot expect it to build an order based colouring transaction. This can happen when a *wallet* containing a mixture of coloured and uncoloured coins is used by a non-colour aware client.

We must also take into account that, even if a client is colour aware, it is possible that it lacks of a certain definition, presumably because the user has no interest on such definition. This would imply that **the client treats a set of coloured coins as if they were uncoloured**.

Apart from the technical issues, we must not forget about the legal issues regarding virtual currency. The latest growth of virtual currency project has also increased governments' and legislator's interest in the topic. Several legislations exist regarding virtual currency issuing, and most of them are not compatible with *one-click money issuing*.

2.2.1.5.1 Colour leaking

When a transaction is built where the **sum of inputs of a given colour is higher than the sum of outputs of that colour coloured coins disappear**. The solution proposed to this is to simply consider that some coloured coins have leaked and live with it. The amount of coins decoloured is equal to the difference of the input and output sums.

2.2.1.5.2 Colour mixing

Whenever an output is the result of the collection of two or more colours it is **impossible to decide what the colour of the output should be**. In that case, the colour is considered to have been lost and a *colour leakage* has taken place.

2.2.1.5.3 Recoloring

However unusual, it is possible that a user colours some already coloured coins. This use case, however strange, should not be considered a bug. It is possible that a user decides to colour some coins, sends some of those to another user and that he decides that **the coins ought to have several colours**. It is possible to have multiple-coloured coins. However, we must keep in mind that those colours cannot be split from the coin.

2.2.1.5.4 Coloured fees

If all the inputs of a transaction are coloured and fees are paid, those fees will be of the colour of the input.

2.2.1.5.5 Legal issues

Finally, legislators do not seem comfortable with non-centralized value exchange. This view seems even stronger when it regards the production of banking tools such as the ones that would appear with the coloured coins. To prevent proliferation of banking products that cannot be controlled[47], there are strong prohibitions and restrictions are imposed around the world[48], such as not allowing BTC to interact with FIAT currencies[49].

2.2.1.6 Alternatives

No matter what the virtues of coloured coins are, they are not the sole and only solution to the production of more complex banking utilities. Several alternatives to coin colouring exist, with their own strengths and weaknesses. Some of them are presented in following sections.

2.2.1.6.1 Stock exchanges

Classical stock exchanges used to work as described below. Several traders would gather in a building and keep track of their transactions.

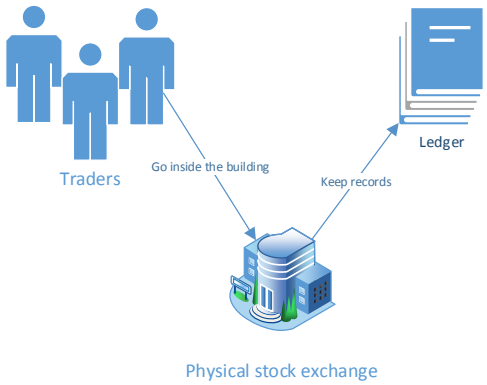

Figure 11 Physical stock exchange

Nowadays there is a worldwide ecosystem of stock exchange where brokers can issue and trade financial instruments. This system has proven to work[50] and be viable for the purpose of *world value exchange*. Stock exchange markets have a significant barrier of entry for small issuers. Moreover, stock exchange does not take advantage of digital representation of assets and require all participants to contact a central authority, the market itself.

All of the above mentioned has high impact regarding transaction fees, availability and even security and turns **stock exchange markets into elitist places** that do not grant access to most of population.

2.2.1.6.2 Digital central services

Digital central services work as described below. Several traders connect to a central server that keeps track of their transactions.

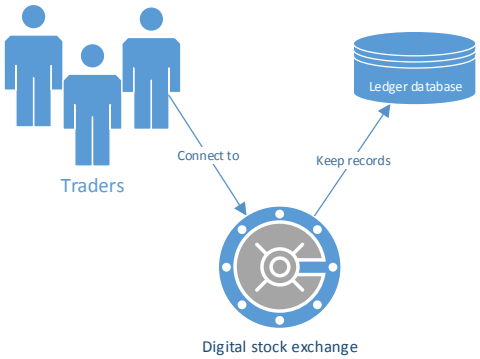

Figure 12 Digital central services

Companies such as GLBSE[51] have tried to take advantage of the Bitcoin power to provide virtual stock markets whose currency is Bitcoin. This offers several advantages when compared to traditional stock exchange markets, such as *international access, anonymity, and lower fees*. However, problems related to central authorities do not go away. The central authority is the *ultimate warrant of the asset ownership of the participants* in the market. So, what happens if the authority disappears? If there is no track record asset owners are thrown to the limbo, they no longer can prove they own their assets, nor can anyone else.

2.2.1.6.3 Dedicated block chain per asset

A system using a dedicated block chain per asset would work as described below. Several traders connect to various *block chain based networks* and sign transactions in those. They would need a means of communication outside the network.

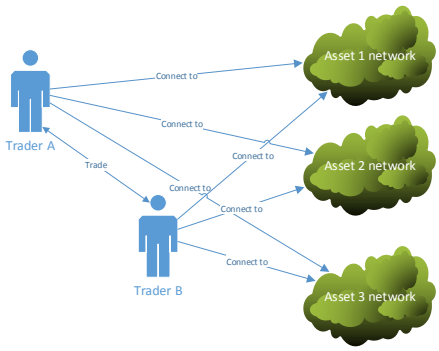

Figure 13 Dedicated block chain per asset

Block chain based networks can be mined simultaneously[52]; this permits the creation of tones of block chain based networks. *Miners* would join merged pools and dynamically chose to mine the networks with most revenue. To issue currency a user might *define his own block chain* and claim that the units of the currency represent a given asset. Software can be written to handle many of these networks and keep track of many of them. Transactions are so malleable that can allow for different block chains to interact without actually having to know much information about it. Transactions can be created so that they are valid in both block chains, so users can trade assets from various block chains. However, this structure has a large entry barrier for small issuers. If a network is of interest to very few users it is expectable that few nodes will serve, store, and verify the data. Also, if all assets are stored in different networks, the client would require several connections to various nodes of each network, **increasing the network overhead**. Also, complex products, such as those requiring *dividend or coupon payments* would become **harder to trace**. It is harder to keep track of tokens of a network that represent *futures* on other networks. If futures are sold in the same network, verification is much easier.

2.2.1.6.4 Asset aware block chain

A system using an asset aware block chain would work as described below. Several traders connect to the *block chain based* and sign transactions in it. They would need a means of communication outside the network.

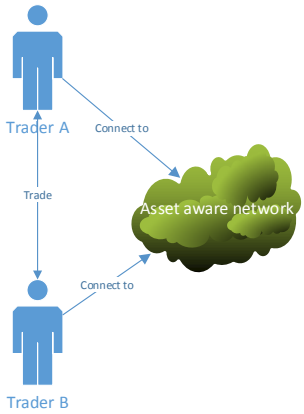

Figure 14 Asset aware block chain

Another approach would be to have an asset, or colour, aware block chain. This block chain could come either in the way of an update on the Bitcoin block chain structure on as a completely new block chain. If the block chain is aware of the colours of the coins it can impose restrictions on the transactions and **only relay the valid ones**. If transactions were coloured, clients could verify that transactions don't create nor destroy coloured coins. Also, the process of **colour verification could be done in constant time** (see Colour aware transactions and block chain); it would not be necessary to perform block scans to recover the colour of transactions.

2.2.1.6.5 Open transactions

The *Open Transactions System*[53] is a system with goals similar to those on Bitcoin. It is also decentralized but *scarcity is based upon signatures of the money issuers*. Also, users do not have to trust any given server, instead *they connect to several servers and ostracise non-honest servers*. Users must communicate between themselves to perform payments and **all the cash instruments are stored in the client side**.

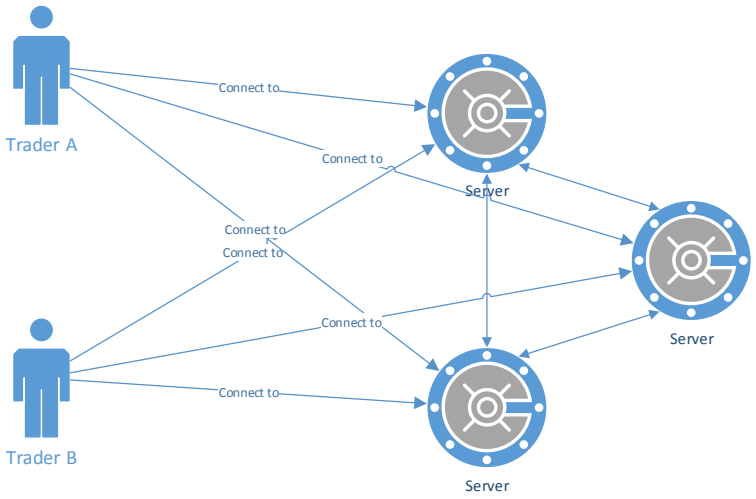

Figure 15 Open transactions

All the communications are secure, in the sense that they cannot be counterfeit. This is achieved by means of asymmetric cyphering of all communication.

Systems like Monetas[54] are possible because they use Open transactions system. This project tries to develop a robust, commercial-grade, fully-featured, and free toolkit implementing OTX protocol[55]. Transactions under this project are based on *strong cryptography*, there is no possible way to modify the balances, and receipts cannot be destroyed and are redundant. Also, there is no way to erase or forge transactions and cash is not traceable.

However, Open Transactions is not built upon the same principles as the Bitcoin network. This probably implies that, if ever, it will be slowly adopted by the Bitcoin community.

3 Description of the solution

The solution proposed (aka **CB2CB**, Coloured Bitcoin to Coloured Bitcoin) adds some functionality to a Bitcoin client. This functionality includes both **distributed exchanging of colour definitions**, as well as **decentralized means for coloured coin exchanges**. The distribution of the colour definition is **based on current torrent protocol**[56], and the **distribution of offers relates to IRC**[57].

3.1 Analysis

In this section we perform an analysis of what it is to be achieved. It is desirable for users to be able to issue and trade coinage in a safe manner. To achieve this, the system must provide the means to keep their anonymity. Coinage counterfeiting must not be possible, and all transactions must be atomically ensured, to prevent fraud. It also must not provide any single point of failure and, preferably, treat dishonest users accordingly.

3.1.1 System architecture

To achieve the goals mentioned above the proposed infrastructure is the one that follows.

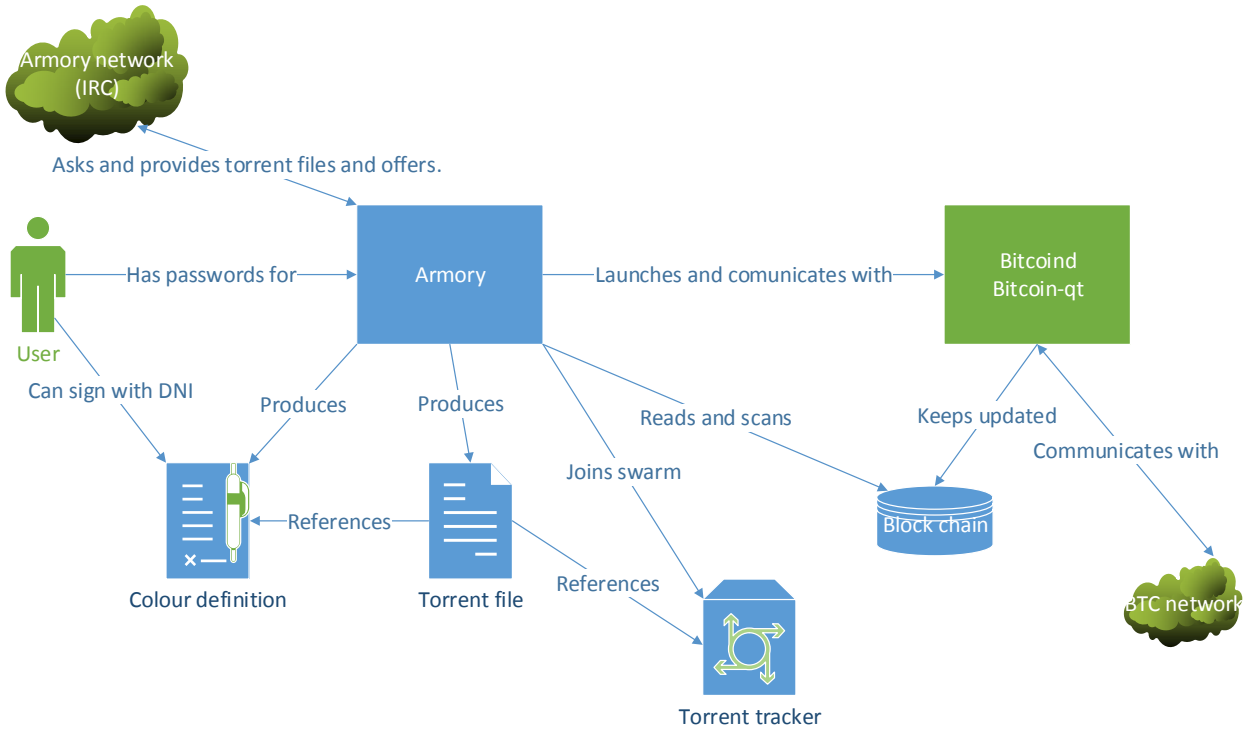

Figure 16 System architecture

It is easy to verify that the *structure presented in Figure 16 provides no single point of failure*. To avoid having to rewrite any code, Amory makes use of the API calls implemented in both Bitcoind[58] and Bitcoin-qt[59] tools. This means that all communication within the Bitcoin Network is done by the Bitcoin client.

Bitcoin software is responsible for connecting to the Bitcoin Network and **keeping the local block chain copy updated**. While Bitcoin software keeps the block chain updated, Armory software can scan it in order keep track of user's accounts. Also, when transactions are to be issued, CB2CB, implemented as an Armory module, creates and sends them to the Bitcoin client, who then relays the transactions. CB2CB is also responsible for the creation of the colour definitions, and generating a torrent file for them. It also provides the means of communication for sharing colour definitions and offers.

3.1.2 Technological study

Here we analyse the possible technologies applicable to the project. Many of them have already been chosen for us by previous developers. Both imposed and eligible technologies will be discussed following.

3.1.2.1 Imposed technologies

There are many technology impositions in the project. All of them come from the software that is to be expanded.

3.1.2.1.1 Armory

This is the Bitcoin client to be expanded. It is coded in two parts. The first one, written in C++, includes all the functionality related to *cryptography*. At the other end of the stack layer, we have a *Graphical User Interface* written in python. *SWIG*[60] technology is used to bind both parts together.

Armory also communicates with the original Bitcoin client (aka Satoshi client) to perform its actions on the Bitcoin Network. This was done to avoid the risk of introducing issues into the Network.

3.1.2.1.1.1 C++

All **cryptographic functions** are written in this language. Then the code gets called from the GUI by means of swig. **CB2CB** will also be coded in this language and will be embedded in the client by the same means.

3.1.2.1.1.2 Python

This language is used to provide the **GUI** as well as the **"glue" for all the modules of the project**. Functionality coded in this language includes both the **communication with the Bitcoin client** as well as **centralized colour definition handling**.

3.1.2.1.1.3 Simplified Wrapper and Interface Generator (SWIG)

It is a tool used to **connect libraries** written in C or C++ with **other languages** such as python, C#, or Java among others. This technology is strictly required duethe various languages used to write the Armory client.

3.1.2.1.1.4 Bitcoin-qt / Bitcoind

Bitcoin-qt is the Satoshi Bitcoin client with a user interface. Bitcoind is an identical piece of software that *runs without a user interface* (aka *daemon*). Both clients support API[61] calls to allow integration with other systems. This is the mechanism Armory uses to broadcast information to the Bitcoin Network.

3.1.2.2 Applicable technologies

3.1.2.2.1 Anonymity provider

From the security point of view, we do not need to cypher our communications because the objective is to make messages impossible to counterfeit, not to prevent users from reading them.

However, for the sake of anonymity, several technologies exists that can provide it, i.e. *tunnelling technologies*. There are two types of tunnelling that would apply great to the case. Each one with its own advantages and disadvantages, discussed below. The use of this kind of technologies is transparent to the client. They have perfect integration with any TCP/IP application layer[62] based protocol.

3.1.2.2.1.1 TOR

This tool provides anonymity by cyphering the message, as well as hiding message issuer's identity. Which ensures almost **impossible traceability** of the communications. However, this technology might **not work as fast a desirably under heavy load**. Still, it should be the **best tool for the average user**.

3.1.2.2.1.2 VPN

VPNs provide anonymity in the sense that all the user's connections appear on the Internet as if they were **established by the VPN exit node**. However, many VPNs **store logs** that allow anyone with access to them to track usage[63]. These tools are usually **not free but provide faster access than TOR[64]**.

3.1.2.2.2 Offer exchanging

It is required to provide the means of offer sharing without a *single point of failure*. This can easily be achieved by almost identical to Internet Relay Chats (IRCs) systems.

3.1.2.2.2.1 Internet Relay Chat

This protocol allows **text messaging using the Internet**. Chats can be built that have no *single point of failure*. This is achieved by having grouped physical servers that relay each other's messages. A similar system can be built to exchange offers.

3.1.2.2.3 P2P sharing

Users must also be able to share colour definition. There are two different approaches to this problem. What the best one is *depends strictly on the size of the definitions that are to be shared*.

3.1.2.2.3.1 Internet Relay Chat

We can use protocols similar to IRC to issue our colour definitions. This would suffice, as long as definitions are short. If definitions are too large the network could get saturated for either the downloader or the uploader.

3.1.2.2.3.2 Libtorrent

Another option is using the current torrent infrastructure in order to share the colour definitions. This would imply that, instead of sharing the colour definitions themselves, users desiring to obtain a given definition would have to ask for the torrent file only.

This solution can be an *overkill* if definitions are small enough, because it is possible to have larger torrent files than definitions themselves. However, due to the malleability of colour definitions, these can become extremely large. If colour definitions contain large amounts of information regarding the coloured coin, which should be expectable, the torrent distribution could be a good solution as distribution of the torrent files would become much more efficient than distribution of the definitions.

Torrent files also include features such as support for multiple trackers, or even providing online references to the file.

Rastebar has a very comfortable implementation of the torrent protocol called **libtorrent-rastebar**[65]. This library allows the *creation of torrent clients with very few instructions*.

3.1.3 Use cases

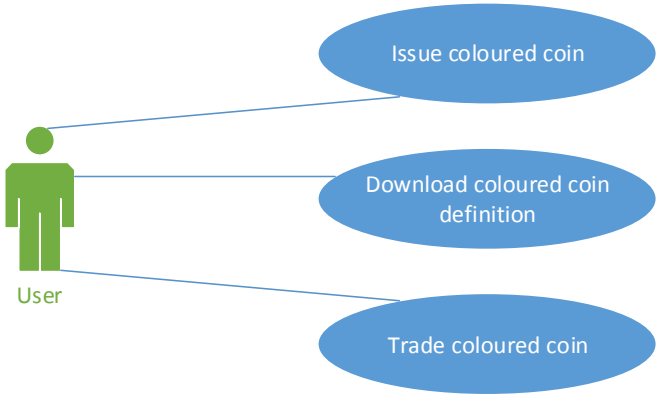

Figure 17 Use cases

Users must be able to **issue coloured coinage**, **download colour definitions** generated by other users, and **trade coloured coinage**.

3.1.4 Software requirements

This section is split into functional requirements and non-functional ones.

3.1.4.1 Functional requirements

ID	Name	Description	Stability	Priority
FR00	Create transaction	The program must be able to generate transactions from the user's accounts.	Low	High
FR01	Generate colour definition	The program must be able to produce colour definitions with the user's preferences.	High	High
FR02	Join torrent tracker	The program must join torrent trackers in order to download other definitions	High	High
FR03	Create torrent file	The program must create torrent files for the definitions it produces.	High	High
FR04	Generate offer	The program must be able to generate offers	Low	High

FR05	Relay offer	The program must be able to relay offers from the network.	High	High
FR06	Sign offer	The program must sign offers, by means of ECDSA.	Low	Low
FR07	Ask for peer list	The program must be able to ask its peers for their respective lists.	Low	High
FR08	Ask for colour definition list	The program must be able to ask its peers for the colour definitions list.	Low	High
FR09	Get colour definition torrent	The program must be able to ask a given peer for a torrent file of a colour definition.	Low	High
FR10	Join peer	The program must be able to discover and join new peers.	Low	High
FR11	Give peer list	The program must provide its peer list when asked to	Low	High
FR12	Give colour definition list	The program must provide its definition list when asked to	Low	High
FR13	Send colour definition torrent	The program must provide a torrent file for the definition	Low	High
FR14				

Table 9 Functional requirements

3.1.4.2 Non-functional requirements

ID	Name	Description	Stability	Priority
FR00	Portability	The software should run in as many platforms as possible.	High	Low
FR01	Secure offers	It must be impossible to forge an offer.	High	High
FR02	Secure definitions	It must be impossible to forge a definition.	Low	High
FR03	Low network usage	The program must use the least amount of network bandwidth.	High	Medium

Table 10 Non-functional requirements

3.1.5 Acceptance tests

ID	Tested elements	Input	Output
AT00	FR00	List of transactions to create	List of transactions created
AT01	FR00 – FR01	Colour definition parameters	Colour definition file, genesis transaction broadcasted
AT02	FR02	Torrent file of a colour definition	Colour definition file
AT03	FR03	Colour definition file	Torrent file of the colour definition
AT04	FR04	Offer parameters	Offer
AT05	FR05	Offer	Offer relayed to other nodes
AT06	FR06	Offer, private key	Signed offer
AT07	FR07	Connect to self, ask for peer list	Received peer list, equal to our own

AT08	FR08	Connect to self, ask for definitions list	Received definitions list, equal to our own
AT09	FR09	Get colour definition	Torrent file obtained
AT10	FR09	Torrent file	Colour definition downloaded
AT11	FR10	IP and port of another node	The two nodes are connected
AT12	FR11	Message requiring peer list	Peer list is sent to the peer
AT13	FR12	Message requiring for colour list definition	Colour list definition is sent over the network
AT14	FR11	Message asking for colour definition torrent	Torrent file sent to peer

Table 11 Acceptance tests

3.1.6 Risks

There are several risks related to the development of the project. Not fulfilling non-functional requirements can have catastrophic consequences. This project, as are those related with money, has a **very deep need for security**. If, by any means, transactions could be forged, or colour definitions altered in any way, scams would start appearing and the software would be *rendered useless*. Because of this, **software will only work on top of the testnet block chain[66], until it is ready for deployment**. This block chain is used as a testbed for Bitcoin related projects.

3.1.7 Legal concerns

Some of the usages of this project might incur in various **legal violations**. Creation of financial products, such as participations, is usually government regulated[67]. For example, UK's government has created an extensive legislation regarding online currencies[68]. This legislation imposes that coin issuers should register with a government's agency and report to it. Also, in other countries such as Spain, even if there is no specific legislation on virtual currency issuing, there are other laws regarding financial products[69].

If software users are **to stay legally safe**, the best solution would be **avoiding all monetary-related coin issuing until they can find a proper lawyer**.

Developers will hold no responsibility, whatsoever, on the usage of the software.

3.2 Detailed design

First we will provide more detailed, but still general, information on how the software structure looks like in different use cases. Then, components responsible for the previously described requirements will be exposed. Functionality will be broken to class level.

3.2.1.1 Colour issuing

Users must be able to issue coinage. This implies that several steps must be taken and that files generated must comply with certain properties.

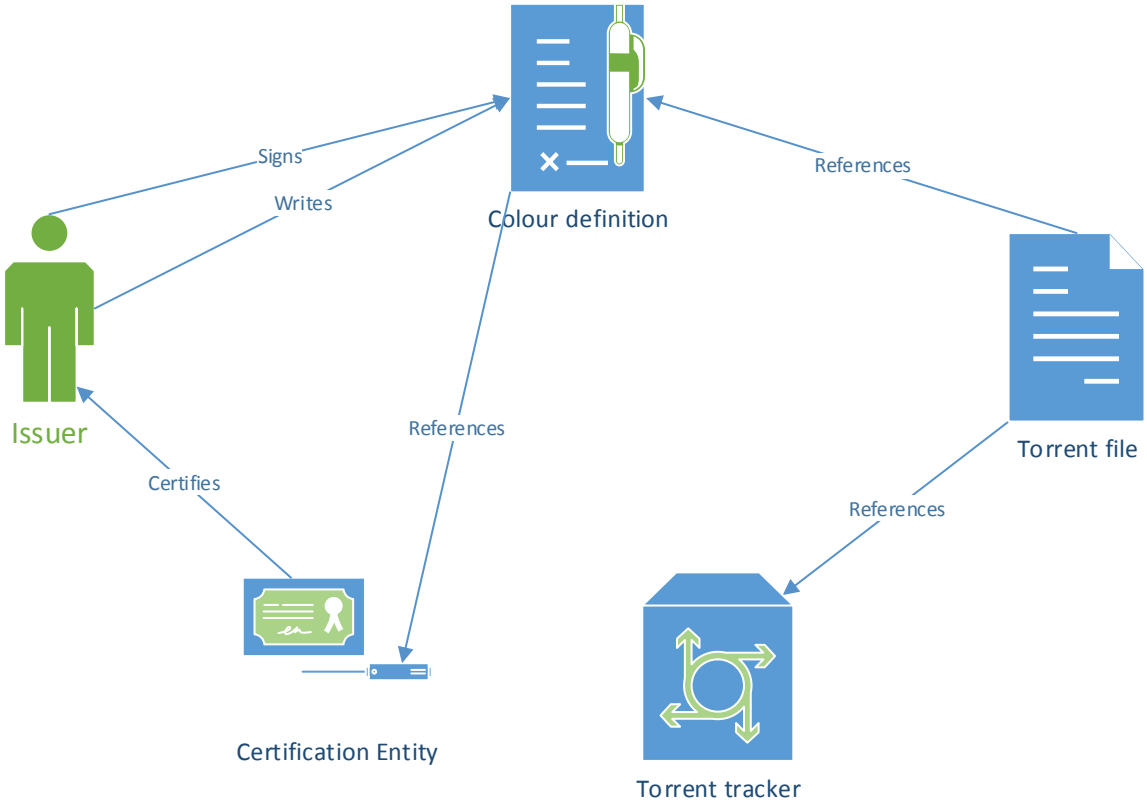

Figure 18 Colour issuing in detail

When an issuer wants to create a colour definition he must **produce the definition file** itself, plus a **torrent file linking to the definition**. Preferably, torrent files will be **added to several trackers**.

3.2.1.2 Colour trading

Users must be able to trade coloured coins amongst themselves. The following figure describes the system when a user is buying coloured coins.

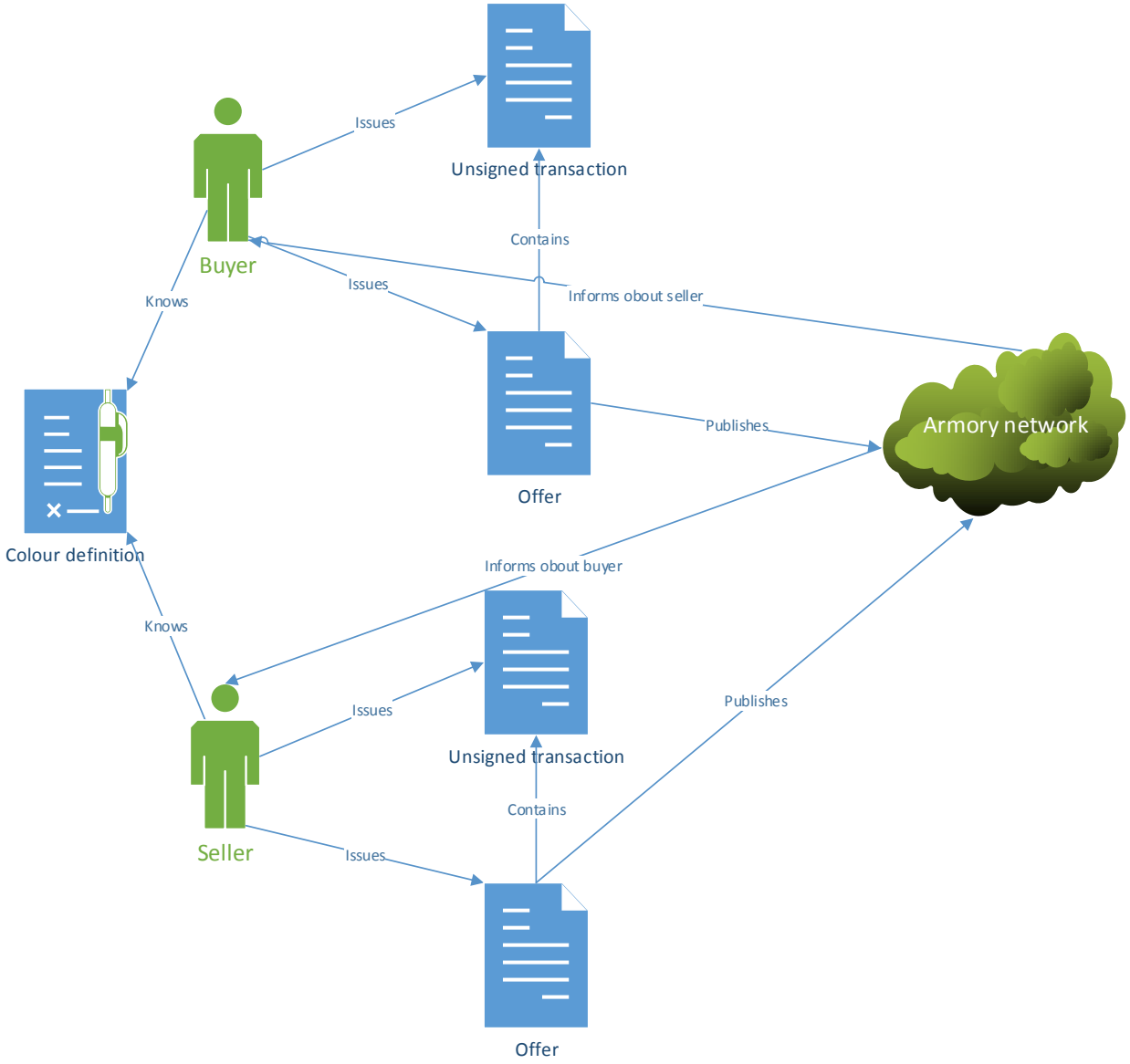

Figure 19 Colour trading in detail

When two users want to trade coloured coins they must **both know** about the **colour definitions involved in their trading**. This means that, if Bob wants to sell some red Bitcoins to Alice, both Bob and Alice must have knowledge of the definition. The unsigned transaction acts as a **proof of ownership** of the coins.

3.2.1.3 Colour definition downloading

Finally, users must have a **secure and decentralized way of sharing their colour definitions**. Luckily, nowadays there is software capable of doing exactly this, the torrent protocol.

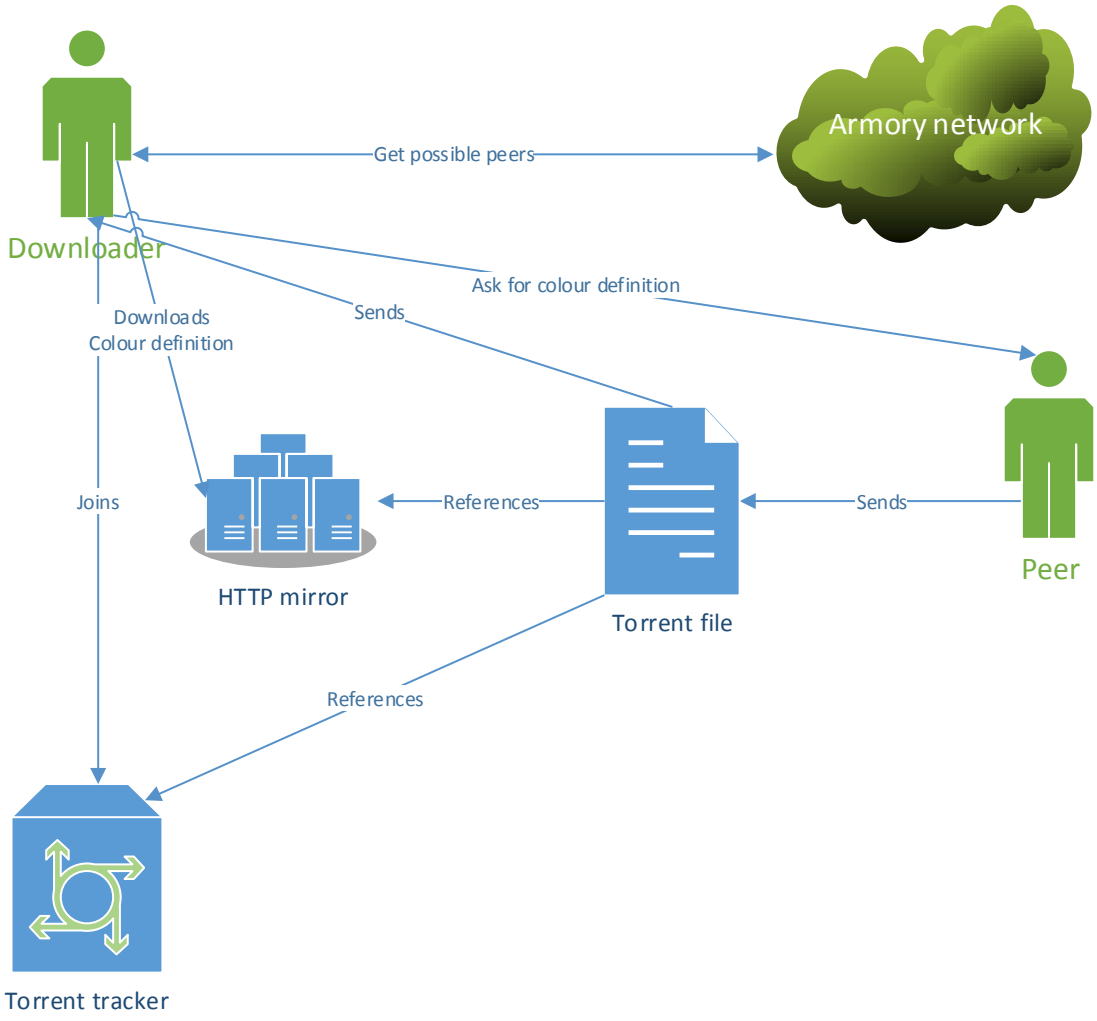

Figure 20 Colour definition downloading in detail

When a user wants to download a certain file, he can either go to some torrent tracking site that contains the torrent for his definition, or ask his peers for it. Because the clients can ask their peers for the torrent file, it is possible to build a client that downloads all the torrent files he can find, and then share them by other means (for example, by creating an *HTTP mirror website*).

Once the downloader has obtained the torrent for the colour definition, he can proceed to download in the same way as any other torrent client would.

3.2.2 Software design

In this section the software design is described by means of UML class diagrams as well as UML sequence diagrams.

3.2.2.1 Class diagrams

Below is the specification of all the classes involved in the project.

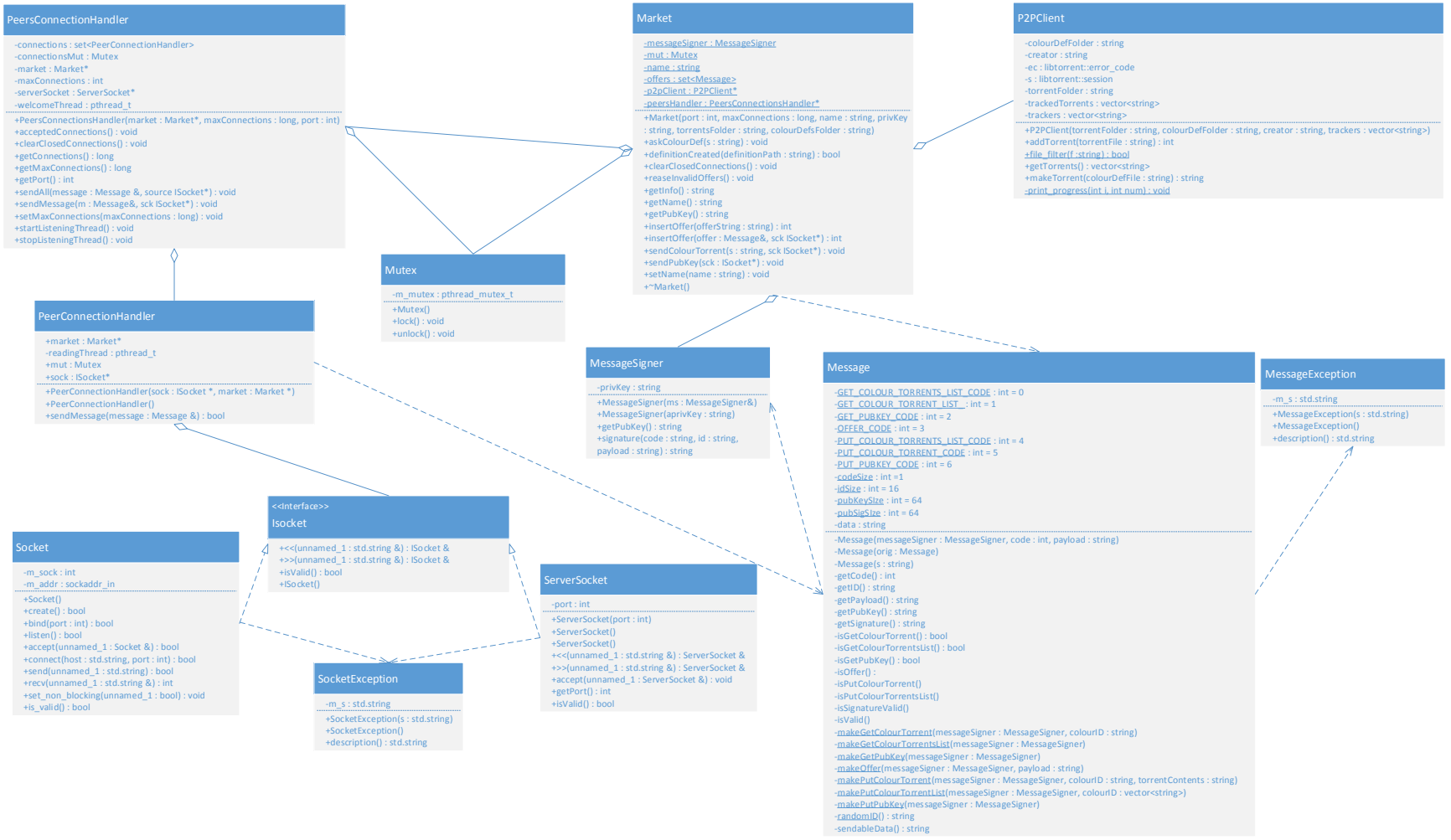

Figure 21 UML class diagram

3.2.2.1.1 Market

Figure 22 Market class

This class behaves like a *marketplace*. Peers are interconnected among them and exchange information through the market. The **Market** is the infrastructure that allows the communication to be useful. It is capable of identifying itself to other markets, by the means of the ECDSA algorithm[70].

3.2.2.1.2 Message

Figure 23 Message class

This class **represents all the exchanged messages among the peers**. It provides static methods to create all kinds of messages.

3.2.2.1.3 MessageSigner

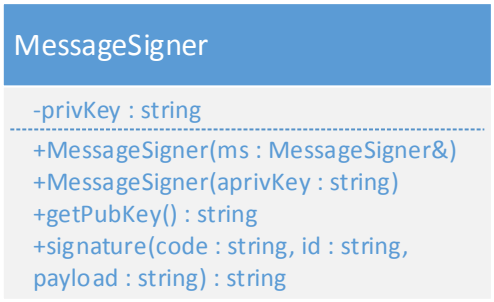

Figure 24 MessageSigner class

This class is responsible for **signing messages**. Each time a message is built it must have access to an object of this class to prevent forgery. This is achieved by means of ECDSA algorithm.

3.2.2.1.4 PeerConnectionHandler

Figure 25 PeerConnectionHandler class

The instances of this class are responsible to **maintain communications with a single peer**. They must **inform the market** that their peer asks for something, while also informing their peer that the market wants something from them. This class can be seen as the *local representative* of the peer.

3.2.2.1.5 PeersConnectionHandler

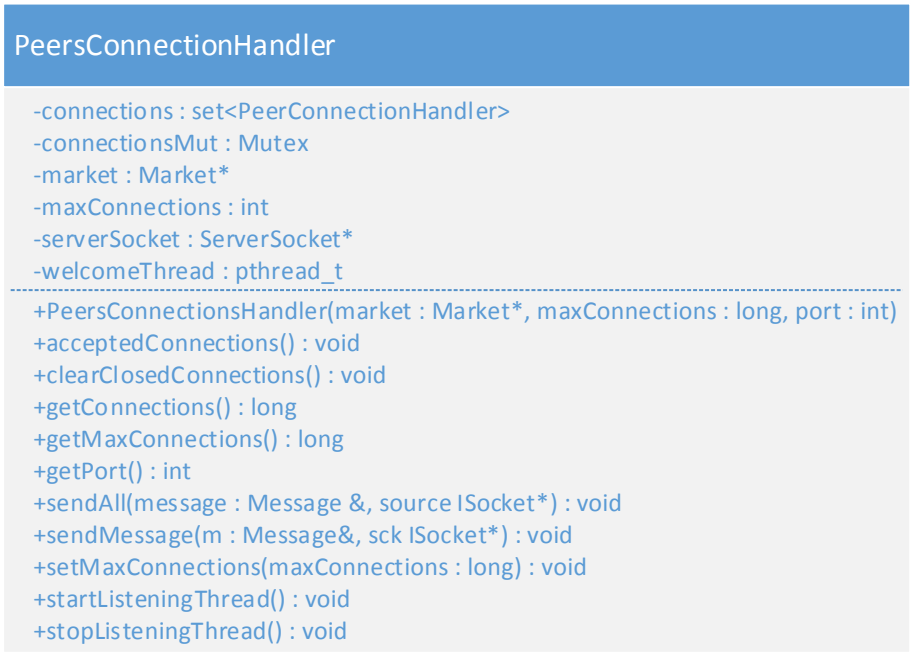

Figure 26 PeersConnectionHandler class

Instances of this class are used to **keep peers logically together**. This class is responsible for the amount of peers connected to the market and also has references to each of them. It **implements the means to start and stop safely all communication**.

P2PClient

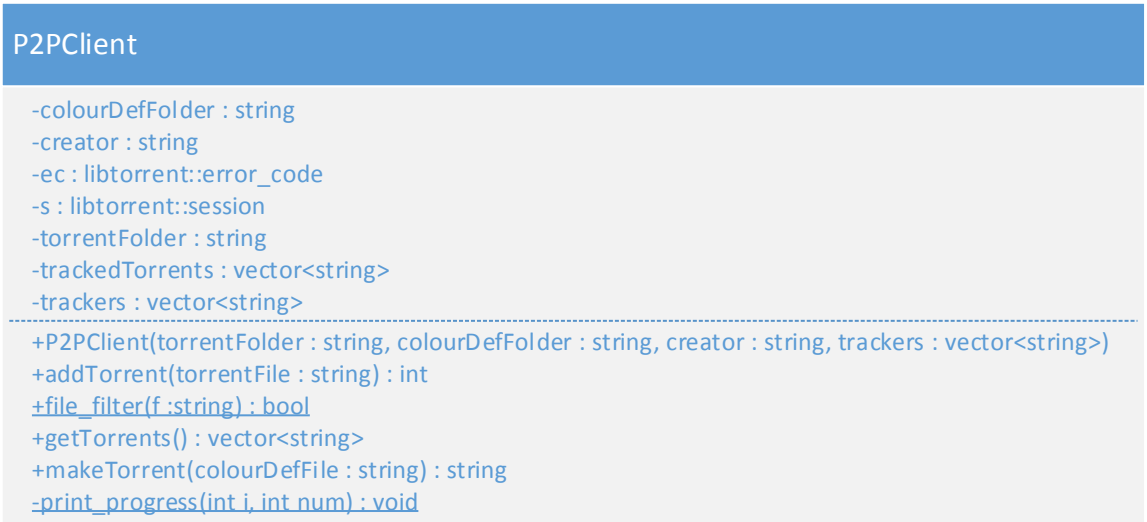

Figure 27 P2PClient class

Instances of this class behave as a **torrent client**. This class is responsible for all the **torrent related work**. This includes creation torrent files when new definitions are created and joining torrent trackers to download desired definitions.

3.2.2.2 Communication structure

The communication structure is based in a P2P network approach. To keep things interesting and in order to prevent isolated sub-networks, users should navigate from one peer to another.

3.2.2.3 Sequence diagrams

Sequence diagrams are interaction diagrams that show entities' interactions in several scenarios. The scenarios described in this section are the same ones as described in the use cases.

In this section we describe the sequence diagrams for each use case. Some cases include asynchronous calls.

3.2.2.3.1 Colour issuing

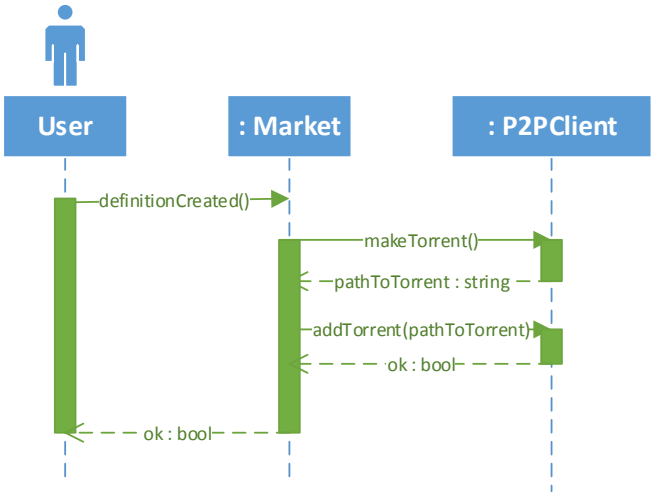

Figure 28 Colour issuing sequence diagram

In this sequence diagram we can observe how users can add their definitions to the system so that they are tracked by the swarm of users. After the sequence is followed any user with the generated torrent can obtain the definition.

3.2.2.3.2 Colour trading

3.2.2.3.2.1 Insert offer

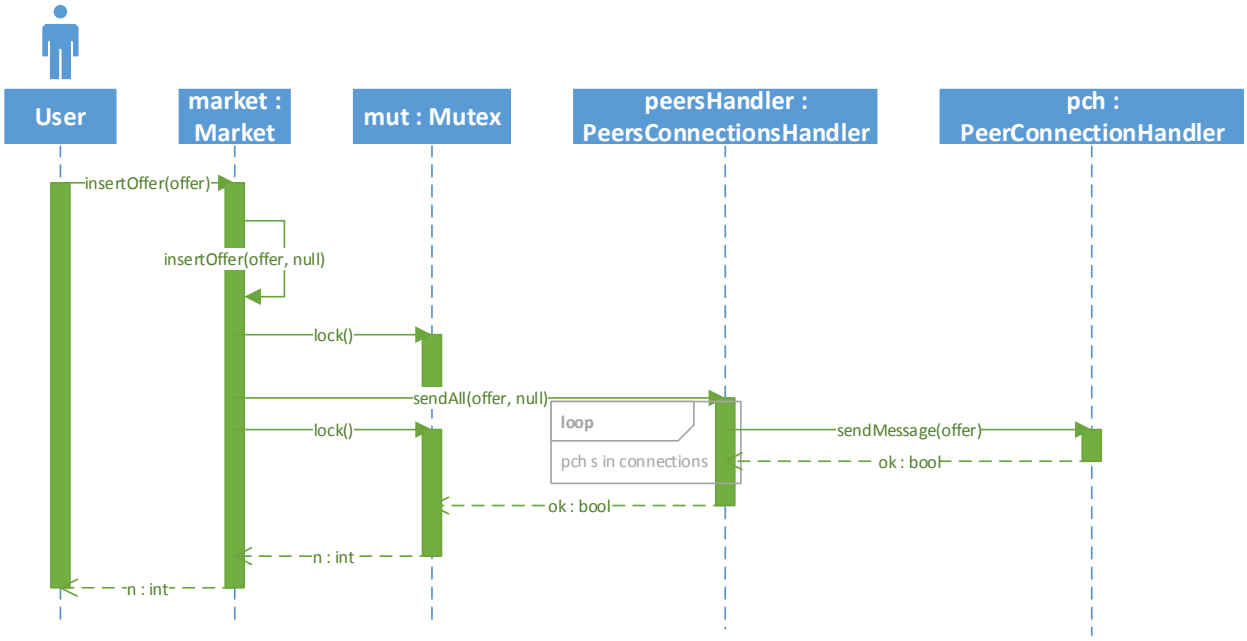

Figure 29 Offer creation sequence diagram

This sequence diagram shows us the flow when a user creates an offer and relays it to his peers.

3.2.2.3.3 Colour definition downloading

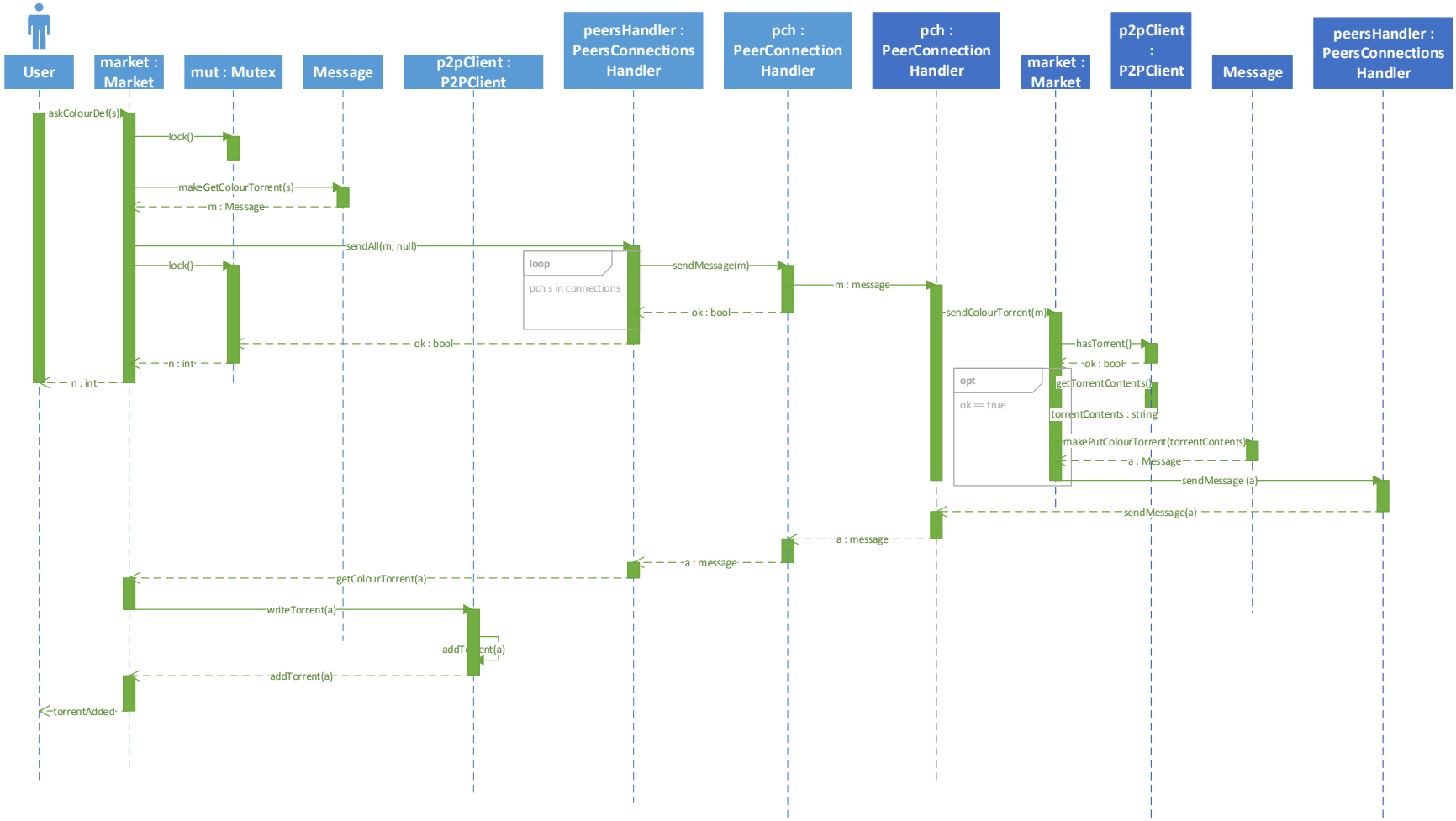

Figure 30 Colour definition downloading sequence diagram.

In this sequence diagram we can observe how users can **obtain colour definitions**. They ask their peers for the definition and, those who have it, send it back. This permits for **adding definitions** from the same network. Also, **torrent files can be hosted on websites**.

Colour definition downloading, asking for information

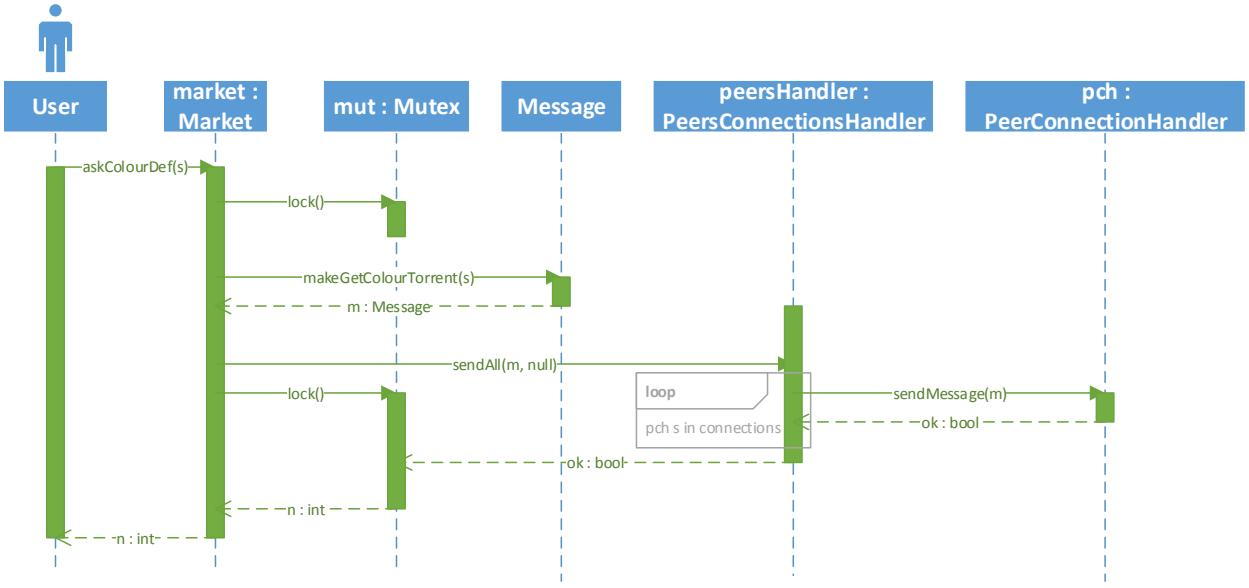

Figure 31 Colour definition downloading, asking for information sequence diagram.

This sequence diagram displays the interactions amongst the objects when a **user asks for a colour definition**. The asynchronous part of the call has been removed.

3.3 Testing

In this section of the document **testing goals**, the **tools used**, and the **results obtained** are described.

3.3.1 Goals

The goals of the assessing process are to ensure that all the use cases can be executed without an error and that they do not have insecure parts.

3.3.2 Testing bench

The software was tested using four virtual machines. All of them with the same software configuration (*Ubuntu 12.04 LTS, 32bits*) and the same virtual hardware, consisting in *1 core processor* and *1 gigabyte of RAM*. The virtualization process was done with *Oracle VirtualBox*.

3.3.3 Results

The results obtained with the acceptance test are detailed below.

ID	Input	Expected Output	Passed
AT00	List of transactions to create	List of transactions created	YES
AT01	Colour definition parameters	Colour definition file, genesis transaction broadcasted	YES
AT02	Torrent file of a colour definition	Colour definition file	YES

AT03	Colour definition file	Torrent file of the colour definition	YES
AT04	Offer parameters	Offer	YES
AT05	Offer	Offer relayed to other nodes	YES
AT06	Offer, private key	Signed offer	YES
AT07	Connect to self, ask for peer list	Received peer list, equal to our own	YES
AT08	Connect to self, ask for definitions list	Received definitions list, equal to our own	YES
AT09	Get colour definition	Torrent file obtained	YES
AT10	Torrent file	Colour definition downloaded	YES
AT11	IP and port of another node	The two nodes are connected	YES
AT12	Message requiring peer list	Peer list is sent to the peer	YES
AT13	Message requiring for colour list definition	Colour list definition is sent over the network	YES

Table 12 Acceptance test results

3.4 Tools used

In this section, the tools used to develop and support the project are described. Both technologies required during the development and documentation phases are detailed. For the sake of brevity, tools such as *compilers*, *linkers*, or *standard libraries* are considered to be provided by the operating system and will not be detailed.

3.4.1 Development tools

Below are described the tools used in the development process. These include software for document production, integrated development environments, file sharing solutions, operating systems, network security, and chatting.

3.4.1.1 Visual Paradigm (UML)[71]

Visual Paradigm is a privative product, with a free community edition, developed by *Visual Paradigm International*. It is a **UML case tool** that provides round-trip engineering for several languages, including C++.

3.4.1.2 NetBeans[72]

This **IDE** can be used for **developing software** written in C++. This makes it very suitable for the development of the code for the project.

The IDE is written in the Java language and was made by *Sun Microsystems*. Later, *Sun Microsystems* made it open source in 2000. In 2010, *Oracle Corporation* acquired *Sun Microsystems*, thus NetBeans.

The version used for the development is 7.3.

3.4.1.3 Dropbox[73]

Dropbox is a **web service based file hosting system**. It allows users to store their files in their cloud, in exchange for their privacy[74]. Also, larger spaces can be acquired. File sharing with other users is also possible.

3.4.1.4 Microsoft Word[75]

Microsoft Word is a privative product developed by *Microsoft*, to be executed using Microsoft Windows. It is a powerful **word processor,** part of the Microsoft Office suite.

3.4.1.5 Microsoft Visio[76]

Microsoft Visio is a privative product developed by *Microsoft*, to be executed using Microsoft Windows. It is a powerful **diagramming tool** that uses vector graphics and is also part of the Microsoft Office suite.

3.4.1.6 Github[77]

Github is a **web-service based application**, and related hosting services, that uses the **git revision control system**[78]. Most of the code used in the project is stored in this hosting service.

3.4.1.7 Ubuntu[79]

Ubuntu is an **Operating System based on the Linux Kernel**. It offers a comfortable open operating system that allows for development of all the required modules of the project.

3.4.1.8 Windows 7

MS Office does not work really well in Linux-based systems. That is the main reason Microsoft Windows 7 **Operating System** was needed in the project.

3.4.1.9 The Onion Router[80]

The Onion Router is a free tool that allows for **online anonymity**. This enables users to browse the web safely, and also permit anonymous connections for all network based software. It works in a way similar to tunnelling through several VPNs (though its tunnelling capabilities reside in *application* layer in TCP/IP network protocol stack instead of *link*, *network* or *transport* layers). All relays only know of the previous and the next hop. The client is the one who builds the path through the network nodes and ciphers its messages so that relays can only see the next hop.

This architecture ensures that, if a node knows a user's IP, they do not know the final destination of the content. If the relay is the output node, it will only know the final destination, and possibly the payload, without any extra knowledge of the source.

3.4.1.10 Pidgin[81]

Pidgin is a **multiprotocol chat client** used to connect via IRC protocol for Internet text messaging exchange. It allows for private communications between users. The **#bitcoin-dev** channel, located at the IRC network *freenode*, which is always full of Bitcoin developers who are really eager to help out.

3.4.2 Client tools

Below are described the tools required to run the client. This are all the required tools to run the client.

3.4.2.1 Python[82]

Python is a **high-level programming language** that emphasizes on **code readability**. This permits for programmers to use fewer lines of code than in other languages. Many programming paradigms are supported by python, such as *imperative, object-oriented, and functional*.

Even if python is a **scripting language**, it can be used in non-scripting contexts. It is possible to pack standalone python code that will execute in a *portable* manner.

3.4.2.2 Bitcoin client[83]

The Bitcoin client is used in order to communicate with the Bitcoin network. The client is capable of API calls, so all the **communication** with Armory is **transparent** to the user.

4 Examples of usage

In this section some examples illustrate the system in use. All of them are fictional and their sole purpose is to show the possible uses of the presented technology. Not only are all the examples possible but they also could be **performed while keeping complete anonymity** and **without a central point of failure**.

4.1 £ Mint

United Kingdom's central bank decides to *issue pounds* in the Bitcoin network in a coloured form.

Because this is a relatively new technology, they agree on a very small issuing. In the description of the coin they produce, users can find proof, by means of certification, that the £ Mint actually belongs to the Bank of England. The definition also contains information on *how to cash back the tokens for physical pounds*. These kinds of colours are very desirable in the Bitcoin network. Bitcoin value fluctuates a lot and that is not good for traders, who prefer stable money.

If these tokens were issued, users would be able to trade pounds inside the Bitcoin Network. They would have the advantages of established currency (**low volatility**) while enjoying the benefits of using the Bitcoin Network (**not being traceable**).

4.2 Teletubby Finances sells a Contract For Difference on BTC~€

Contracts For Differences (CFDs) are contracts between two parties, usually identified as *Buyer* and *Seller*. CFDs are derivative financial products that permit traders to take advantage of prices going up and down. *CFDs specify that the seller will pay the buyer the difference between the present value of the asset and the value when the contract was created*. If the difference were negative, the buyer would pay it the seller.

CFDs can have limited risk. Traders can put boundaries on minimum and maximum prices. If this were the case, *Teletubby Finances* could sell CFDs for the boundaries and *act as mediator*.

If CFDs are sold at the maximum loss to each party, users can freely trade them, with the warranty behind that *Teletubby Finances* will return the corresponding payments to each party.

4.3 Warlord territorial expansion

This example can be a bit cruel so two versions of the same story have been written. The hard version is completely **not recommended** for sensitive readers[84].

For this example we should better be in a fantasy world.

The inhabitants of *Colour Flatland* are polygons whose segments are coloured. As polygons get old, their edges begin to fade; but they can cannibalise other polygons' edges to stay alive a little longer.

Edge reusing is very much like human blood transfusions. Some colours are good for more polygons than others; they have to potential to help more polygons.

There is a zone in *Colour Flatland*, *Candyplace*, where polygons are very rich, but they are growing old, and have a desperate need for new edges. However, they are much civilised and do not desire to cannibalise their own kind.

Far from *Candyplace* there is a warlord, *Megagon,* which is planning to attack a nearby village. He has been so much time fighting that he has almost lost all of his resources. So, after a council, he agrees to put some financial imagination into practice. *Megagon* knows that the village he plans to attack has 30 young polygons. The council agrees that they can sell the polygons' edges.

First, they issue 30 tokens representing each of the young polygons. However, and to keep investors safe, CFDs on the polygon colours are also issued. CFDs state that *Megagon* will get to keep more money if the captured edges are of the most valuable colours. Also, CFDs will be sorted out, by means of multiple-signature transactions, by *Dr Square*, a beloved doctor from *Candyplace*.

After all the coins are set up, *Megagon* sells them in packs including the same amount of CFDs and polygon edges' tokens. From now on there is an open market on the edges and their CFDs.

Once the attack is successful, *Dr Square* comes into the scene and checks the colours of the captured polygons. After this, transactions are updated to reflect reality; finally payments are made and young polygons' edges are ready to be harvested.

4.4 Mining operation participation (aka mining bonds)

A Bitcoin *miner* decides to take investors in his operation. To attract investors he issues mining bonds[85] by means of coloured coins. He builds a colour description including a description and proof of his operation. Also, he provides an account that will be *the holder* of the mined Bitcoins, and decides to issue 100 bonds. Every coin mined by the provided account will be split evenly with the bond holders.

It is easy to keep track of the payments done by the *miner*. He will, undoubtedly, have included the specifications of his mining operation in the colour description. These must include the *hashing power he owns*, so predictions, and deviations from them, can easily be found. Other users *might create other financial instruments on top of these bonds*.

4.5 Bitcoin laundering

Companies are set up to *launder peoples' money*. In the Bitcoin Network this is achieved by what is known as **coin swapping**. If Alice and Bob have "*dirty*" Bitcoins, they can trade those in order to hide where they came from. Usually, large pools of Bitcoins are made, so that many transactions can be created, thus *diluting the money's traceability*. Companies are set up to help build these swarms and create these transactions[86].

4.5.1 How it works without colouring

Currently, if Bob wants to launder his 10 Bitcoin, he has to go to a company responsible for a large enough pool. The company will take his Bitcoins, put them in their mixing pool, and finally, after some time, send them to one of Bob's accounts, minus a fee.

The *more mixing that is desired, the longer the laundering process takes*. However, the longer it takes, the longer Bob will have to wait for his Bitcoins. *Once the process is started Bob has no access to his coins.*

4.5.2 How it could work if colouring was introduced

Money laundering pools could issue coloured coins for their users. Each coin could have its own definition, be ciphered so that only Bob can read it. When Bob sends his 10 Bitcoin to the laundering pool, he obtains a *coloured coin, with a cyphered definition that only he can read*. At any time, Bob can go to the pools' web site and see when his coin will mature. Not only can he do this, but he can also show his friend Alice the deciphered definition, allowing her to check for that maturity. Now that Alice knows when the laundered coins will be available, she can safely buy the coin from Bob for 9 BTC.

5 Conclusions and future lines of work

As exposed along the document, Coloured Bitcoin *is really promising*. However, the network was not designed to support colour operations and this turns them into a relatively high cost operation (O(n)). If Bitcoin transactions supported colour tagging, users would benefit from a network that, in the worst case scenario, has a constant cost. This seems more desirable than implementing colouring on top of the current network.

However, Bitcoin structure was designed to be hard to modify. When a change is introduced, *miners* have to decide if they wish to accept the change and, in case they don't, the change simply cannot take place.

In the previously described scenario, and as a temporal solution, a system such as the one described could permit societies to live "*as if*" having banks, but without them.

To develop the project it was required to use multiple heterogeneous technologies. Also, much has been learned on how cutting-edge monetary systems work.

5.1 Future lines of work

Before any more functionality is added to CB2CB, it is extremely important to test it thoroughly. This should include, but no limit to, **peer source code revisions** and formal **analysis of the source code**.

When dealing with software oriented to be **bank-grade secure**, a lot of testing is unavoidable.

5.1.1 Rate tokens

Tokens, as they are *cheaply made coins*, could also be used as *rating tokens*. Companies could issue coloured coins containing a message of certain colours, as well as the rating they give to that colour. Information stored in the block chain is really expensive (*0.005 BTC per kilobyte of information*). However, rating companies might still find it *profitable to provide public ratings of the colours*.

5.1.2 Swarm markets

Groups of users interested in trading a certain colour could join trackers of such colour. Very much like a torrent tracker, users could join lists and swarms of peers *with their very own interests*. This should allow for *higher efficiency and scalability of the offer trading system*.

5.1.3 Market policies

At the moment of writing this document, *clients relay all the offers*, and *do not fee users in any way for it*. This is not what should be expected from a market. Also, offer relaying can be very political; some users might want to say: "*I will only help to make transactions in which markets do not make money*". To cover this use case, clients should implement *relaying policies* and make them public to other clients. This would prevent users from connecting to clients who will never relay their offers.

Transactions have fees, this means that some kind of markets are *not viable*. On the one hand, those markets that require of thousands of transactions each second (aka *exchange markets*) need lower fees to be viable. On the other hand, traditional markets exist, where users go, buy what they need, and then go away. These type of markets have been named as *common markets*.

5.1.3.1 Exchange market

An exchange market, as exposed above, is *a market that is capable of thousands of transactions per second*. An online market can be built so that it stores a copy of all transactions and that, periodically, reduces the transactions and broadcasts this reduction to the network.

These types of markets require *some kind of security* ensuring that users will not collect their transactions before the reduction takes place. This security can be obtained, for example, by *buying tokens required to operate in the market*. If a user creates an illicit transaction, the value of his token is lost and he has to *rebuy an entrance token* to the market.

5.1.3.2 Common market

Common markets can be open to everybody because *there is no trust required in them*. Transactions are performed atomically, require a fee, and might take some time before they are accepted in a block.

5.1.4 Migration to Electrum

Electrum is a new Bitcoin client that no longer stores all the block chain, achieved by using a *remote server* that allows it to run on small devices, such as smartphones. Migration to such a system might be worthy because of the benefits of *smaller storage* and *lower initial loading time*.

5.1.5 More robust colour definitions

At the present time, users can create definitions of transactions they are completely unrelated with. This is not a good idea. On the bright sight, colour definitions are JSON files and these are extremely malleable. It should be fairly easy to only accept colour definitions that are signed by the account that collected the inputs of the transaction. If more than one account did so, clients should expect to find all the required signatures. This would *ensure that no one creates a colour definition for coinage he has no control of*.

5.1.6 Semantic implications of coloured Bitcoins

Each Bitcoin colour can represent an existent object or concept in the real world. So this real concept matching against colours can be seen as semantic information. It would be possible to develop a system relating the exchange rates to defined colours in the Bitcoin Network. Colours' exchange rate can be obtained in a similar fashion as currency exchange rates are evaluated. We can see this system as one with no central reference (*virtual barterer-trade*).

6 Annex 1: Glossary

API	An Application Programming Interface is a specification on how some software should interact with another. It can refer to routines, data structures, and objects.
Armory	Armory is an open source Bitcoin wallet manager. It provides high level security, while still being usable. It covers all basic needs of users, and it can be used in standard or advanced mode.
ASIC	Application-Specific Integrated Circuits are integrated circuits designed for a particular use. They provide better consumption than general purpose circuits.
Bitcoin Testnet	The Bitcoin Testnet is an alternative block chain that is used for testing. It permits developers and testers to experiment without having to worry about breaking the bitcoin chain.
Block chain	A block chain is a transaction database that is shared among the network nodes. It behaves as a ledger of all the previous transactions and makes double-spending almost impossible.
BTC	The term BTC is used to refer to the Bitcoin coins. At the time of this writing, the acceptance of it by ISO 4217 is still pending[87].
Coin swapping	Coin swapping is the term used to refer situations where Bob sends Alice the same money that Alice sends Bob, usually to make the coins untraceable.
Daemon	A daemon is a program that runs as a background process. It is not under the direct control of an interactive user.
DNS	The Domain Name System is the system used to associate information to domain names. Its primary function is to translate domain names to IP addresses.
ECDSA	Elliptic Curve Digital Signature Algorithm is a variant of DSA based on Elliptic Curve Cryptography.
Elliptic Curve Cryptography	Elliptic curve cryptography is used in the public-key cryptography field. It uses elliptic curves over finite fields to provide.
Futures	When using financial terminology, a future is a type of contract where tow traders agree on a future price[88].
Genesis block	The genesis block is the first block of the block chain.
Block chain fork	Block chain forks represent disagreement of the miners. A split of the ledger occurs and users follow either one branch or the other.
Hash	A hash is a message digest produced by a cryptographic hashing function, such as SHA256.
Hashing power	Hashing power is the capacity to compute hashes.
HTTP mirror website	Mirrors are exact copies of data. Precisely, an http website mirror is a mirror that shares its data as a web service.
Merged mining	When a miner decides to mine, at least, two block-chains and does so by mining both at the same time, it is said that he is merged mining.
Merkle tree	A Merkle tree, also known as hash tree, is a tree built using hashes. Leaves of the tree are plain data blocks, that later get combined by

	hash nodes.
Miner	A bitcoin miner is a user that computes block hashes in order to solve a block and claim the reward and the transaction fees.
Mining bond	A bond issued by a miner, usually to finance his operation, where he sells the coins he will produce.
OTX protocol	OTX is an open protocol that provides secure exchange of value among various users.
Portable application	A portable application is an application that does not have any external requirements.
Proof of work	Proof of work are hard to solve and easy to verify problems. They are used as a deter measure to prevent denial of service attacks or spam.
Quantum computing	Quantum computing refers to both quantum computers and quantum algorithms. Both are based on superposition and entanglement properties of quantum-mechanical phenomena to represent data with such properties.
Rating token	Rating tokens are issued by rating agencies. They provide a public means of communication to all the users of a particular block chain.
SHA256	SHA256 is a cryptographic hashing function designed by the U.S. National Security Agency and published in 2001.
SSL	Secure Socket Layer, and Transport Layer Security (TLS) are cryptographic protocols to provide secure communications over the Internet.
Stack automaton	A pushdown or stack automaton is an automaton that uses a stack. It is more capable than a finite-state machine but not as powerful as a Turing machine.
SWIG	Simplified Wrapper and Interface Generator is an open source tool that is used to connect computer programs and libraries written in C or C++ with many scripting languages.
TOR	The Onion Router is free software that provides anonymity on the Internet. It allows users to circumvent surveillance, traffic analysis, and censorship.
BitTorrent	BitTorrent is a protocol to support P2P file sharing. It is used to distribute large amounts of data over the internet and is considered to be responsible for 43%-70% of all the Internet traffic[89].
Tunnel	IP tunnels are used in network communications to provide channels between two networks. They are often used to connect two disjoint IP networks without a native path to each other.
VPN	Virtual Private Networks extend private networks across a public one. They establish virtual point-to-point connections between the two networks that are wished to join.
Wallet	Wallets are users' files that contain a collection of private keys. They permit the spending of the user's owned coins.

7 Annex 2: Project management

This annex contains the specifications of the various aspects related to the project management. These include the work planning, the technical means required and the economic analysis of the project.

7.1 Planning

In this section we detail the original planning and the real development, as well as the deviations.

7.1.1 Initial planning

This section details the initial planning of the project. Various phases have been defined and time has been estimated for all of them. In the elaboration of the project 8 hour journeys have been taken into account.

The planning goes from the 7th of January to 17th of June. This adds up to a total of **97 working days**.

The table below shows the detailed planning.

Task name	Duration	Begin	End
Final degree project	**117 days**	**07/01/13**	**18/06/13**
Initial planning	**3 days**	**07/01/13**	**09/01/13**
State of the art	**30 days**	**10/01/13**	**20/02/13**
Study of the Bitcoin network	15 days	10/01/13	30/01/13
Study of colour issuing	15 days	31/01/13	20/02/13
Analysis	**23 days**	**21/02/13**	**25/03/13**
System architecture	3 days	21/02/13	25/02/13
Technological study	7 days	26/02/13	06/03/13
Use cases	2 days	07/03/13	08/03/13
Requirements definition	6 days	11/03/13	18/03/13
Acceptance tests	5 days	19/03/13	25/03/13
Design	**16 days**	**26/03/13**	**16/04/13**
Software design	12 days	26/03/13	10/04/13
Sequence diagrams	4 days	11/04/13	16/04/13
Implementation	**15 days**	**17/04/13**	**07/05/13**
Testing	**10 days**	**08/05/13**	**21/05/13**
Documentation	**20 days**	**22/05/13**	**18/06/13**

Table 13 Detailed initial planning.

As we can see in the table previously showed, the development fits a cascade model. This is because there is only one worker, which does not want to do the same work more than once. Below we have a figure of the Gantt diagram for the initial planning

Nombre de tarea	Duración	Comienzo
Final degree project	**117 días**	**lun 07/01/13**
Initial planning	3 días	lun 07/01/13
State of the art	**30 días**	**jue 10/01/13**
Study of the bitcoin network	15 días	jue 10/01/13
Study of colour issuing	15 días	jue 31/01/13
Analysis	**23 días**	**jue 21/02/13**
System architecture	3 días	jue 21/02/13
Technological study	7 días	mar 26/02/13
Use cases	2 días	jue 07/03/13
Requirements definition	6 días	lun 11/03/13
Acceptance tests definition	5 días	mar 19/03/13
Design	**16 días**	**mar 26/03/13**
Software design	12 días	mar 26/03/13
Sequence diagrams	4 días	jue 11/04/13
Implementation	15 días	mié 17/04/13
Testing	10 días	mié 08/05/13
Documentation	20 días	mié 22/05/13

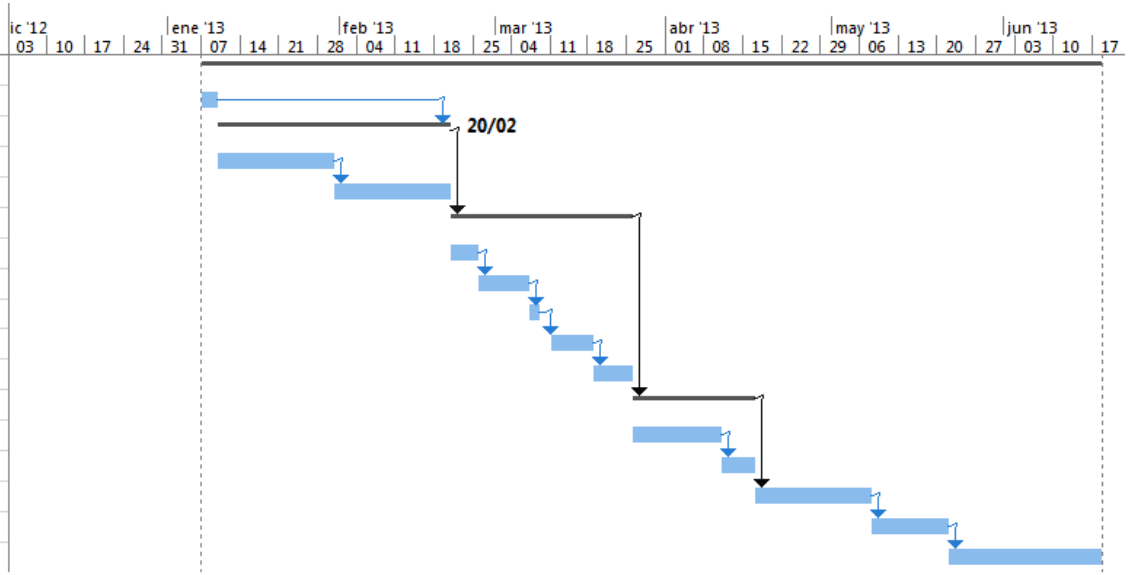

Figure 32 Gantt diagram for initial planning

Real development

In this section we provide information related to the execution of the project. We can observe some *deviations* from the original planning. Time was saved during the analysis and design phases; however, some of it had to be invested to finish the implementation phase. Documenting the work done also took less time than expected.

The table below shows the detailed execution of the project.

Task name	Duration	Begin	End
Final degree project	**101 days**	**07/01/13**	**27/05/13**
Initial planning	**3 days**	**07/01/13**	**09/01/13**
State of the art	**30 days**	**10/01/13**	**20/02/13**
Study of the Bitcoin network	15 days	10/01/13	30/01/13
Study of colour issuing	15 days	31/01/13	20/02/13
Analysis	**14 days**	**21/02/13**	**12/03/13**
System architecture	3 days	21/02/13	25/02/13
Technological study	5 days	26/02/13	04/03/13
Use cases	2 days	05/03/13	06/03/13
Requirements definition	1 days	07/03/13	07/03/13
Acceptance tests	3 days	08/03/13	12/03/13
Design	**12 days**	**13/03/13**	**28/03/13**
Software design	8 days	13/03/13	22/03/13
Sequence diagrams	4 days	25/03/13	28/03/13
Implementation	**17 days**	**29/03/13**	**22/04/13**
Testing	**10 days**	**23/04/13**	**06/05/13**
Documentation	**15 days**	**07/05/13**	**27/05/13**

Table 14 Detailed project execution

It can be seen that the **total amount of days** required to execute the project was **101**. This represents a *deviation from the original planning of 13.68%*. This deviation is, probably, due to the *limited experience* on project planning.

Nombre de tarea	Duración	Comienzo
Final degree project	**117 días**	**lun 07/01/13**
Initial planning	3 días	lun 07/01/13
State of the art	**30 días**	**jue 10/01/13**
Study of the bitcoin network	15 días	jue 10/01/13
Study of colour issuing	15 días	jue 31/01/13
Analysis	**23 días**	**jue 21/02/13**
System architecture	3 días	jue 21/02/13
Technological study	7 días	mar 26/02/13
Use cases	2 días	jue 07/03/13
Requirements definition	6 días	lun 11/03/13
Acceptance tests definition	5 días	mar 19/03/13
Design	**16 días**	**mar 26/03/13**
Software design	12 días	mar 26/03/13
Sequence diagrams	4 días	jue 11/04/13
Implementation	15 días	mié 17/04/13
Testing	10 días	mié 08/05/13
Documentation	20 días	mié 22/05/13

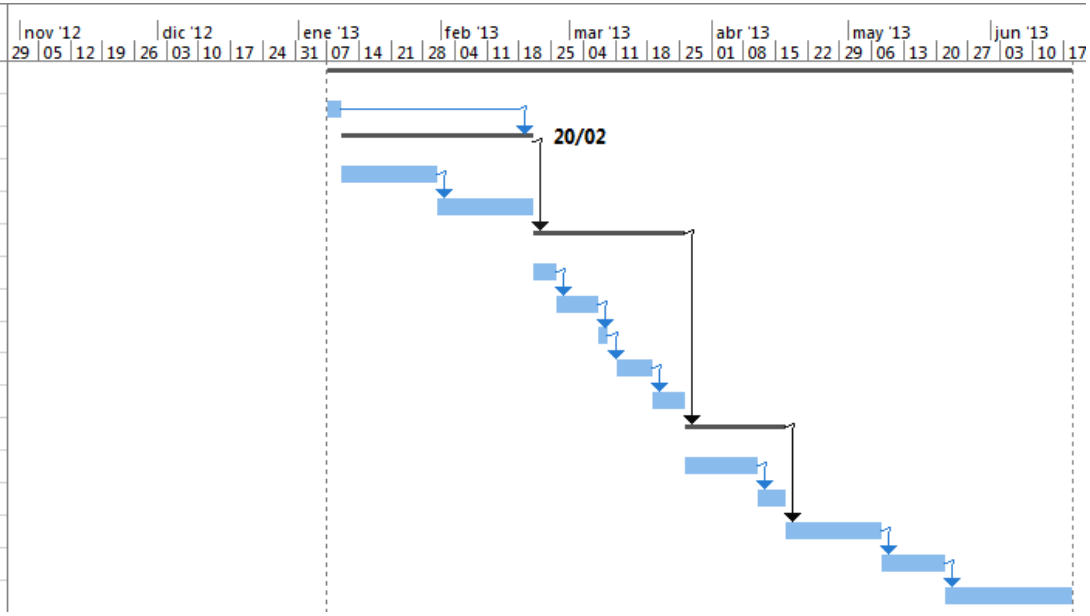

Figure 33 Gantt diagram for project execution

7.2 Technical means used

In this section the means required for the project execution are described. These have been split between **software** and **hardware** tools used. Both hardware and software tools are detailed below.

7.2.1 Software means

Below we have a table describing the software tools used.

Tool	Description
Netbeans 7.3	This free Integrated Development Environment (IDE), developed by Sun Microsystems, provides a great environment for C++ projects.
Ubuntu 12.04 LTS x86	Ubuntu is a common Linux distribution.
Dropbox	Is a free cloud file storage solution. It allows for file synchronization among computers and also has a web interface.
Windows 7 x64	This operating system will is used to write all the documentation. It is a requirement to use other Microsoft's or Apple's products to run the Microsoft Office suite.
Iron Browser	This is an open source browser based on chromium. It behaves like chrome browser but does not report your navigation data to Google.
Microsoft Office 2013	This is the office suit used to document the project.
Microsoft Visio 2013	This tool from Microsoft is used to create charts and diagrams.
Microsoft Project 2013	This tool from Microsoft is used to create the project's planning.
TOR 0.2.2.39	This free tool was used to provide anonymity while browsing the web.
Pidgin	This free tool was required to join IRC chats and communicate with Bitcoin developers.
Github	This service was used to obtain the initial code on which to work on.
Visual Paradigm Enterprise 10.1	This tool was used to try to automate the creation of UML diagrams.
Zotero	This browser pluggin was used to keep track and present online references.

Table 15 Table with software means

7.2.2 Hardware means

Below we have a table describing the hardware tools used.

Tool	Description
Computer	An i7 965, 24 GB of RAM, 7200 rpm SATA hard drive. This is the computer where all the development has taken place. Documentation was also produced in this station.
Router	Netgear's WNDR3800 router was used to access the Internet.

Table 16 Table with hardware means

7.3 Economic analysis of the project

This section includes the economic analysis of the project. It describes the *methodology used to estimate the costs, initial budget, client's budget, and real cost of the project*.

7.3.1 Cost estimation methodology

Cost estimation has been done taking into account both direct and indirect costs.

Direct costs relate to concepts directly related to the project's development. These include *equipment*, *software*, *wages*, and *traveling costs*. The Spanish Social Security information on general regime contracts has been considered to do the computation. Other costs will be estimated from public prices.

Indirect costs are the ones not directly related to the project development. They include *phone*, *light*, *gas*, *Internet connection* and any *renting the used building* might have. We have estimated these costs as a 20% of the direct costs.

7.3.2 Initial budget

Here the initial budget is described. All of the amounts, unless otherwise stated, are included *without taking VAT into account*.

7.3.2.1 Wages budget

To compute wages expenses we have considered a *single engineer's salary* for the length of the project. We have considered a salary of 20 €/hour, and then added the 23.6% for the Spanish Social Security.

Resource	Dedication	Gross cost	Average monthly cost	Social Security	Total
Engineer	936 h	18,720.00 €	3,528.57 €	4,417.92 €	23,137.92 €
				Total:	**23,137.92 €**

Table 17 Table with detailed wages budget

7.3.2.2 Equipment budget

In this section we detail the expenses related to equipment used during the development of the project. The hardware used has been previously described. The table below is used to describe the costs attributable to the project.

To compute the cost attributable to each product has been computed considering a 36 months depreciation period. The period is so short because it refers to computer equipment.

Product	Cost	Dedication	Depreciation period	Cost attributable
Computer	1138.59 €	117 days	1080 days	123.34 €
Router	185.28 €	117 days	1080 days	20.07 €
			Total:	**143.41 €**

Table 18 Table with detailed equipment budget

7.3.2.3 Software budget

This section details software expenses, as specified in the previous sections. The table below shows the costs attributable to the project. As with the previous calculations, a *36 months depreciation period* has also been taken into account. This is the same one as the one the public administration uses[90].

Product	Cost	Dedication	Depreciation period	Cost attributable
Netbeans	None	117 days	1080 days	-

Ubuntu	None	117 days	1080 days	-
Dropbox	None	117 days	1080 days	-
MS Windows 7 x64 Pro.	40 €	117 days	1080 days	4.33 €
Iron Browser	None	117 days	1080 days	-
Microsoft Office 2013	319.99 €	117 days	1080 days	34.67 €
Microsoft Visio 2013	699.95 €	117 days	1080 days	75.82 €
Microsoft Project 2013	700.00 €	117 days	1080 days	75.83 €
TOR	None	117 days	1080 days	-
Pidgin	None	117 days	1080 days	-
Github	None	117 days	1080 days	-
Visual Paradigm Enterprise	1048.18 €	117 days	1080 days	113.55 €
			Total:	**304.2 €**

Table 19 Table with detailed software budget

7.3.2.4 Consumables budget

This sections details the expected expenses on consumables. This budget covers expenses derived from *office material*; this includes things such as *pens*, *paper*, and so on.

Product	Cost per unit	Amount	Total
Office material	23.13 €	1	23.13 €
		Total:	**23.13 €**

Table 20 Table with detailed consumables budget

7.3.2.5 Travel and per diem budget

This section details budget allocated for travel and *per diem allowances*.

Product	Cost per unit	Amount	Total
Public transport	63.70 € / month	3 months	191.10 €
Per diem	12.5 € / day	117 days	1,462.50 €
		Total:	**1,653.60 €**

Table 21 Detailed travel and per diem budget

7.3.2.6 Direct costs

This section describes the direct costs of the project. The total comes from the sum of the concepts previously described. The table below shows the sum for each of them, as well as the total direct costs.

Concept	Cost
Wages	23,137.92 €
Equipment	143.41 €
Software	304.2 €
Consumables	23.13 €

Travel and per diem	1,653.60 €
Total:	**25,262.26 €**

Table 22 Table with direct costs

7.3.2.7 Indirect costs

As described in the section Cost estimation methodology, indirect costs will be estimated as a 20% of direct ones.

According to this, the indirect costs would be **five thousand fifty-two euro and forty-five cents (5,052.45 €)**.

7.3.2.8 Estimated costs

Concept	Cost
Wages	23,137.92 €
Equipment	143.41 €
Software	304.2 €
Consumables	23.13 €
Travel and per diem	1,653.60 €
Indirect costs	5,052.45 €
Total:	**30,314.71 €**

Table 23 Table with estimated costs

7.3.3 Client budget

This section shows the budget for the client. As mentioned before, a *20% of risk and a 20% of benefits are assumed*.

Concept	Amount
Total cost	30,314.71 €
Risk (20%)	6,062.94 €
Benefits (20%)	6,062.94 €
Total prior VAT:	**42,440.59 €**
VAT (21%):	8,912.52 €
Total:	**51,353.11€**

Table 24 Budget for the project

7.3.4 Deviation analysis

This section details the final cost of the project, as well as the deviation from the initially estimated. The project did not last as expected and this changes both the direct and the indirect costs.

Below there is a table with the deviation from the original budget.

Concept	Budget	Final cost	Deviation
Wages	23,137.92 €	19,973.76 €	-3,164.16 €

Equipment	143.41 €	123.80 €	-19.61 €
Software	304.2 €	262.60 €	-41.60 €
Consumables	23.13 €	23.13 €	0.00 €
Travel and per diem	1,653.60 €	1,453.60 €	-200.00 €
Total direct	**25,262.26 €**	**21,836.89 €**	**-3,425.37 €**
Total indirect	5,052.45 €	**4,367.38 €**	**-658.07 €**
Total	**30,314.71 €**	**26,204.27 €**	**-4,110.44 €**

Table 25 Detailed deviation

The deviation of the cost over the project is -4,110.44 €, a -13.56%. The time invested in the project was less than expected, this **reduces both the direct and indirect costs** of it.

8 Annex 3: User manual

This annex contains the user manual of the application. It is split in two parts. The first one details the preconditions required to compile and run the software. The second one, on the other hand, contains detailed instructions on how to perform the various tasks using the application's interface.

This user manual documents the usage of the added functionality. For the rest of options, please refer to the original manual[91].

8.1 Minimum requirements

The minimum requirements to run the software are to use a Linux system that is capable of running python's User Interfaces, have an Internet connection capable of UPnP, and 32 bit system with a minimum of 256 MB of RAM. However, because only one system has been tested (Ubuntu 12.04 LTS) it is highly recommended.

8.2 Preconditions

The preconditions required to execute the software are split into two categories, both of them have to be met in order to use the software.

8.2.1 Hardware

All the software has been tested on top of virtual machines. However, by looking at the specifications, we can determine that it will work on top of most of the commodity hardware. In fact, a computer as cheap as Raspberrypi B can also run it.

8.2.2 Software

Software requirements vary on whether the user desires to compile the application or simply execute it.

8.2.2.1 Compiling

8.2.2.2 Executing

Both Bitcoin-qt (version 0.7.2), and python are required to properly execute the Armory with CB2CB support.

8.3 Execution and functionality

In this section we will show the users how they might take advantage of the software. It is explained in detail how most of the functionality is accessible from the interface.

Below is an image of the main interface of the software. All of the new functionality can be find in the Halluciante menu. The name has remained unchanged as a sign of respect towards the creator of the original software.

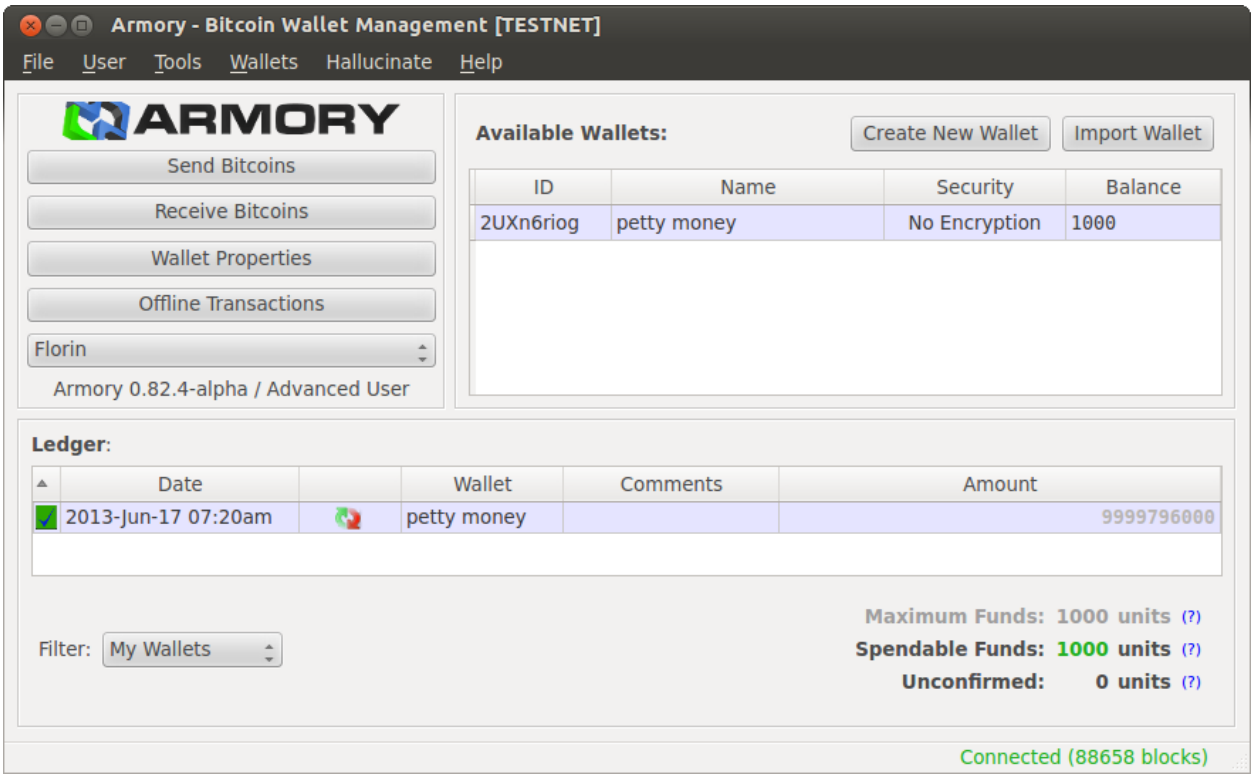

Figure 34 Armory

8.3.1 Colour issuing

Here we see how the issuing interface looks when a user is issuing a coin named Florin.

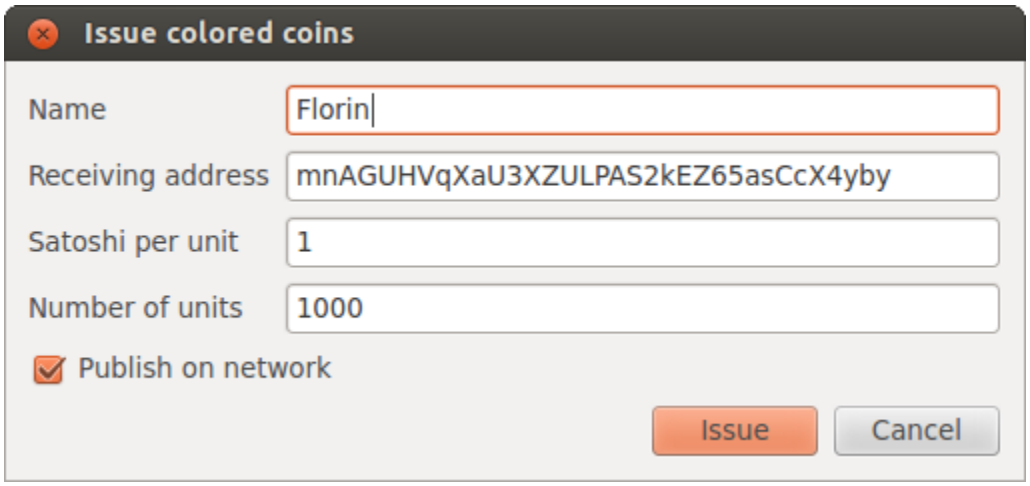

Figure 35 Colour issuing interface

After the user has clicked issue, he is prompted with a message box confirming the issuing transaction.

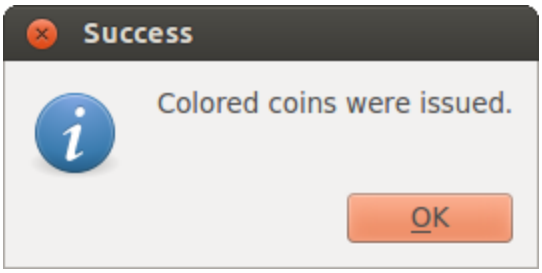

Figure 36 Success message

Once the transaction has been issued, the client creates a tracker of the definition and joins the network for other users to obtain it. Once the colour definition is in the network the user is prompted with such information.

Figure 37 Colour definition uploaded

8.3.2 Colour trading

To trade coins of a certain colour such coins have to be selected in the main interface. Then the user must open the P2P exchange tab where he will be asked to connect to the network.

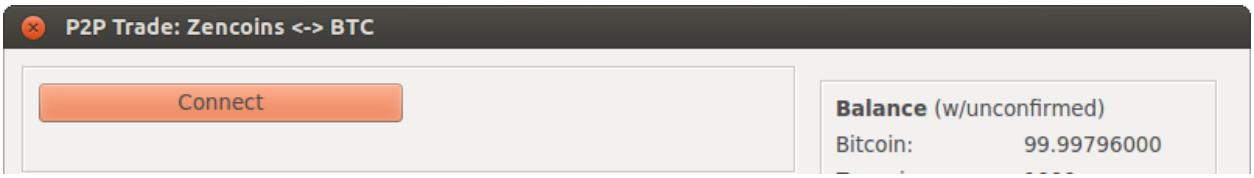

Figure 38 Connect button

Once the user has joined the network, he can issue his offers and these will be relayed amongst the network, allowing other users to match the offer.

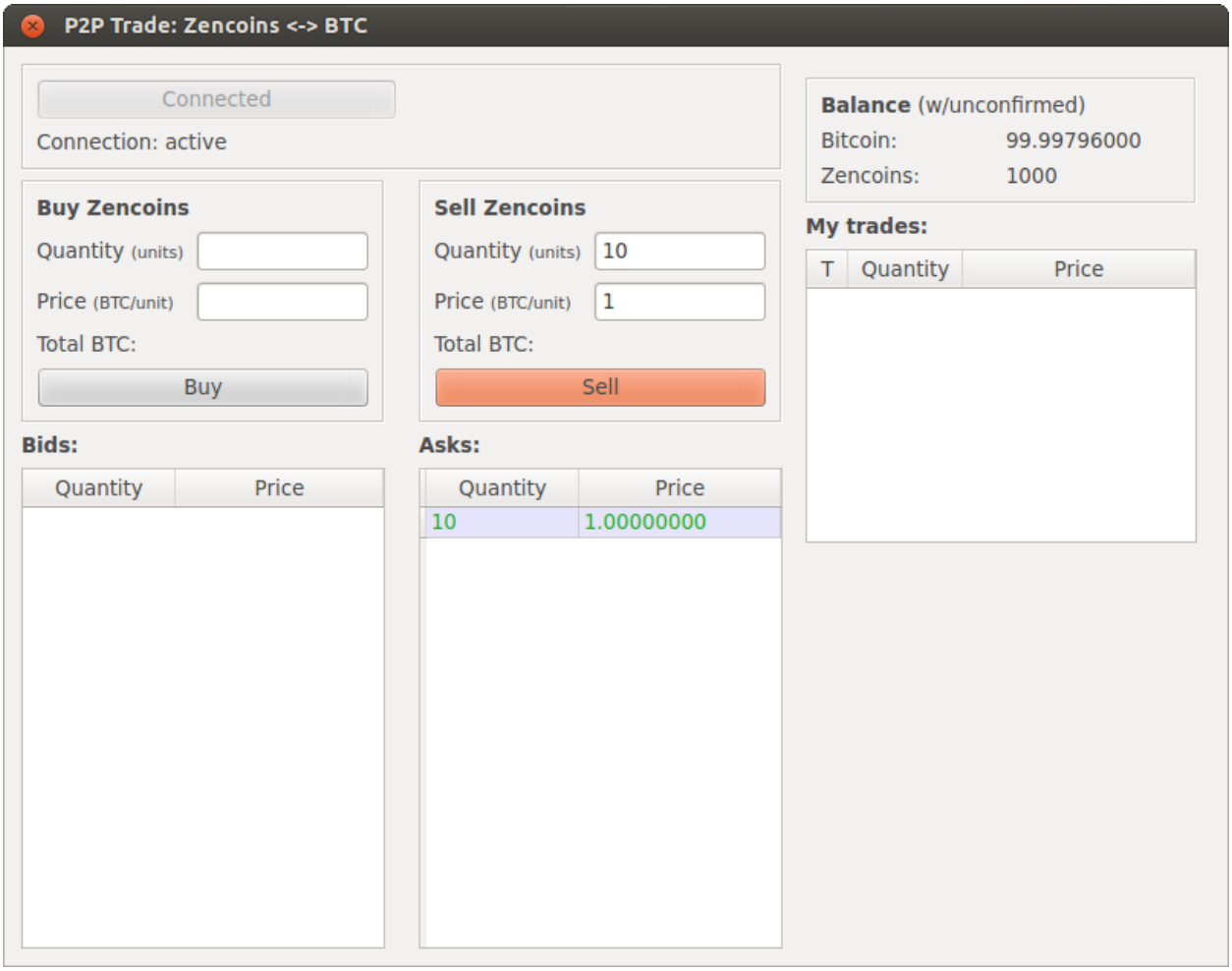

Figure 39 Trading interface

8.3.3 Colour definition downloading

Users can also download definitions. This can be done in two ways. They can either provide an URL of the definition they desire to obtain, or they can provide the colour id and the network will try to find it.

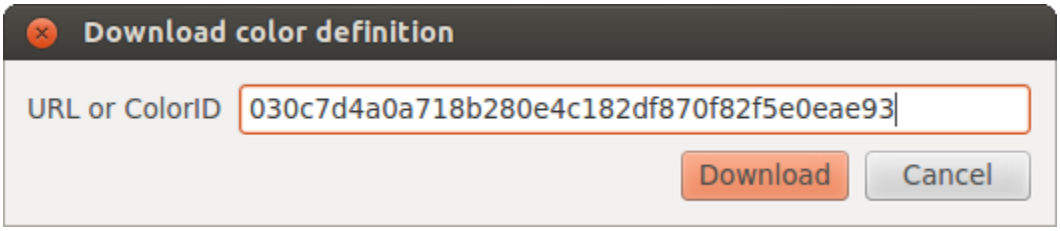

Figure 40 Colour definition downloading interface

As soon as the definition is downloaded, by either of the methods, the user is prompted with a message box asking him if he agrees to install the definition. If he does so the definition is added to the software and will be used after a restart of the system.

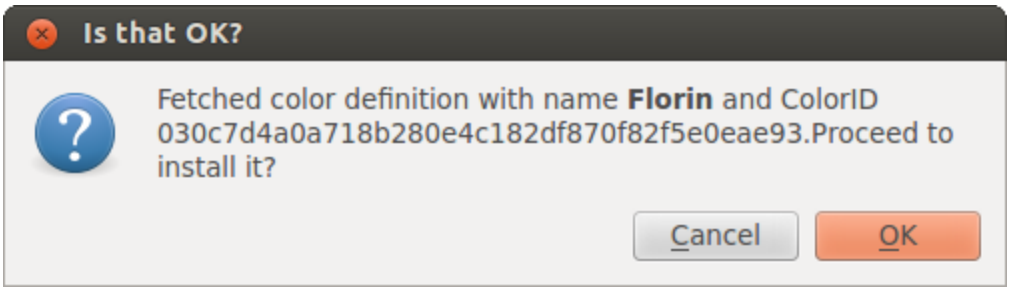

Figure 41 Colour definition acceptance interface

8.4 Other considerations

Users' must take into account that CB2CB is not ready for deployment. Its use on the main Bitcoin network is completely discouraged. However, when using it on Testnet, users have nothing to lose so they should be encouraged to try as much of the functionality as possible.

9 References

[1] BitcoinX Project | Colored Bitcoin P2P Cloud Exchange [WWW Document], n.d. URL http://www.bitcoinx.org/ (accessed 6.23.13).

[2] Colored Bitcoins - BitcoinX - Google Drive [WWW Document], n.d. URL https://docs.google.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/ (accessed 6.23.13).

[3] "Bitcoin: A Peer-toPeer Electronic Cash System", Satoshi Nakamoto. May 24, 2009

[4] "Introduction to Modern Cryptography", Jonathan Katz and Yehuda Lindell. August 2007

[5] Transactions Verification - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Transactions#Verification (accessed 6.23.13).

[6] "Island Money", Michael F. Bryan. Federal Reserve Bank of Cleveland, February 1, 2004.

[7] Introduction Sending payments - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Introduction#Sending_payments (accessed 6.23.13).

[8] TorifyHOWTO [WWW Document], n.d. URL https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO (accessed 6.23.13).

[9] Difficulty - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Difficulty (accessed 6.23.13).

[10] Block Reward Halving: A Guide | Bitcoin MagazineBitcoin Magazine [WWW Document], n.d. URL http://bitcoinmagazine.com/block-reward-halving-a-guide/ (accessed 6.23.13).

[11] ECB: Credit institutions subject to the Eurosystem's minimum reserve requirements [WWW Document], n.d. URL http://www.ecb.int/mopo/implement/mr/html/credit.en.html (accessed 6.23.13).

[12] Block hashing algorithm - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Block_hashing_algorithm (accessed 6.23.13).

[13] Reference to miner cashing fees

[14] Script - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Script (accessed 6.23.13).

[15] Script Words - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Script#Words (accessed 6.23.13).

[16] Script - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Script (accessed 6.23.13).

[17] Protocol specification - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Protocol_specification#Variable_length_integer (accessed 6.23.13).

[18] Units - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Units (accessed 6.23.13).

[19] Blocks - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Block#How_many_blocks_are_there.3F (accessed 6.23.13).

[20] RSA Laboratories - Recent Improvements in the Efficient Use of Merkle Trees: Additional Options for the Long Term [WWW Document], n.d. URL https://www.rsa.com/rsalabs/node.asp?id=2003 (accessed 6.23.13).

[21] Bitcoin network graphs [WWW Document], n.d. URL http://bitcoin.sipa.be/ (accessed 6.24.13).

[22] Litecoin - Open source P2P digital currency [WWW Document], n.d. URL http://litecoin.org/ (accessed 6.23.13).

[23] Patent US4309569 - Method of providing digital signatures - Google Patents [WWW Document], n.d. URL http://www.google.com/patents/US4309569 (accessed 6.23.13).

[24] Transaction fees Relaying - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Transaction_fees#Relaying (accessed 6.23.13).

[25] Namecoin DNS - DotBIT Project [WWW Document], n.d. URL https://dot-bit.org/Main_Page (accessed 6.24.13).

[26] BitDNS and Generalizing Bitcoin [WWW Document], n.d. URL https://bitcointalk.org/index.php?topic=1790.msg28696#msg28696 (accessed 6.23.13).

[27] Avalon ASIC Chips | Asic Chips [WWW Document], n.d. URL http://www.asic-chips.com/ (accessed 6.23.13).

[28] Blocks Description - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Blocks#Description (accessed 6.23.13).

[29] Bitcoin War: The First Real Threat to Bitcoin? | Privacy Online News [WWW Document], n.d. URL https://www.privateinternetaccess.com/blog/2012/03/bitcoin-war-the-first-real-threat-to-bitcoin/ (accessed 6.23.13).

[30] *The FSA's role under the Electronic Money Regulations 2011.* Finantial Service Authority, March, 2011.

[31] Accredited Standards Committee X9, American National Standard X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 16, 2005.
[32] *Quantum Computation and Quantum Information*. p. 202. Nielsen, Michael A.; Chuang, Isaac L. .
[33] Post-quantum cryptography [WWW Document], n.d. URL http://pqcrypto.org/ (accessed 6.23.13).
[34] Home - Bitcoin Block Explorer [WWW Document], n.d. URL https://blockexplorer.com/ (accessed 6.23.13).
[35] Webcoin [WWW Document], n.d. URL http://bitcoinx.github.io/webcoinx/ (accessed 6.24.13).
[36] Armory - Simple. Secure. Solid. [WWW Document], n.d. URL https://bitcoinarmory.com/ (accessed 6.24.13).
[37] Smart Property - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Smart_Property (accessed 6.24.13).
[38] Bitcoin Block Reward Halved to 25BTC | PC Perspective [WWW Document], n.d. URL http://www.pcper.com/news/General-Tech/Bitcoin-Block-Reward-Halved-25BTC (accessed 6.23.13).
[39] U.S. Bureau of Engraving and Printing - The Production Process [WWW Document], n.d. URL http://moneyfactory.gov/uscurrency/theproductionprocess.html (accessed 6.23.13).
[40] DailyTech - Inside the Mega-Hack of Bitcoin: the Full Story [WWW Document], n.d. URL http://www.dailytech.com/Inside+the+MegaHack+of+Bitcoin+the+Full+Story/article21942.htm (accessed 6.23.13).
[41] Multiple block chain transaction
[42] The Money-ness of Bitcoins by Nikolay Gertchev [WWW Document], n.d. URL http://lewrockwell.com/orig14/gertchev1.1.1.html (accessed 6.23.13).
[43] OBC - Order Based Coloring [PDF Document], n.d. URL https://bitcoil.co.il/BitcoinX.pdf (accessed 6.24.13).
[44] Alternative chain - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Alternative_Chains (accessed 6.24.13).
[45] Merged Mining - Namecoin DNS [WWW Document], n.d. URL https://dot-bit.org/Merged_Mining (accessed 6.24.13).
[46] Order Based Coloring in a Dust-Free World - Grupos de Google [WWW Document], n.d. URL https://groups.google.com/forum/#!topic/bitcoinX/r8iCJUMA5xw (accessed 6.24.13).
[47] The tulipmania: Fact or artifact? Earl Thompson, 2007.
[48] Financial Regulation [WWW Document], n.d. URL http://europa.eu/legislation_summaries/budget/l34015_en.htm (accessed 6.24.13).
[49] Bitcoin Valuation as a Fiat Hedge With Information and Substitution Effects | Mineforeman [WWW Document], n.d. URL http://mineforeman.com/2013/05/14/bitcoin-valuation-as-a-fiat-hedge-with-information-and-substitution-effects/ (accessed 6.24.13).
[50] Stock market crash of 1929 (American history) -- Encyclopedia Britannica [WWW Document], n.d. URL http://global.britannica.com/EBchecked/topic/566754/stock-market-crash-of-1929 (accessed 6.23.13).
[51] GLBSE - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/GLBSE (accessed 6.23.13).
[52] Merged mining specification - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Merged_mining_specification (accessed 6.23.13).
[53] FellowTraveler/Open-Transactions · GitHub [WWW Document], n.d. URL https://github.com/FellowTraveler/Open-Transactions (accessed 6.13.13).
[54] Monetas [WWW Document], n.d. URL http://monetas.net/ (accessed 6.23.13).
[55] Open Transactions – Freecode [WWW Document], n.d. URL http://freecode.com/projects/open-transactions (accessed 6.24.13).
[56] BitTorrentSpecification - Theory.org Wiki [WWW Document], n.d. URL https://wiki.theory.org/BitTorrentSpecification (accessed 6.23.13).
[57] RFC 1459 - Internet Relay Chat Protocol [WWW Document], n.d. URL https://tools.ietf.org/html/rfc1459.html (accessed 6.23.13).
[58] Bitcoind - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Bitcoind (accessed 6.23.13).
[59] Bitcoin-Qt - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Bitcoin-qt (accessed 6.23.13).
[60] Simplified Wrapper and Interface Generator [WWW Document], n.d. URL http://swig.org/ (accessed 6.23.13).
[61] Original Bitcoin client/API calls list - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Api (accessed 6.23.13).

[62] "Computer Networking: A Top-Down Approach", James F. Kurose, Keith W. Ross. Pearson Education, Limited, 2012
[63] Can Commercial VPNs Really Protect Your Privacy? | Techdirt [WWW Document], n.d. URL https://www.techdirt.com/articles/20130402/02421422545/can-commercial-vpns-really-protect-your-privacy.shtml (accessed 6.24.13).
[64] TOR vs. VPN [WWW Document], n.d. URL http://torguard.net/blog/tor-vs-vpn/ (accessed 6.23.13).
[65] Libtorrent n.d. URL http://www.rasterbar.com/products/libtorrent/ (accessed 6.23.13).
[66] Testnet - Bitcoin [WWW Document], n.d. URL https://en.bitcoin.it/wiki/Testnet (accessed 6.23.13).
[67] Financial Regulation: The Concise Encyclopedia of Economics | Library of Economics and Liberty [WWW Document], n.d. URL http://www.econlib.org/library/Enc/FinancialRegulation.html (accessed 6.23.13).
[68] Real legal issues with virtual currencies [WWW Document], n.d. URL https://www.networkworld.com/newsletters/2010/051010sec2.html (accessed 6.23.13).
[69] CNMV - Comisión Nacional del Mercado de Valores [WWW Document], n.d. URL http://www.cnmv.es/portal/home.aspx (accessed 6.23.13).
[70] The Elliptic Curve Digital Signature Algorithm (ECDSA). Don Johnson, Alfred Menezes and Scott Vanstone, Certicom Resarch, Canada
[71] Visual modeling tool for building enterprise applications [WWW Document], n.d. URL http://www.visual-paradigm.com/product/vpuml/features/ (accessed 6.23.13).
[72] Welcome to NetBeans [WWW Document], n.d. URL https://netbeans.org/ (accessed 6.23.13).
[73] Dropbox [WWW Document], n.d. URL https://www.dropbox.com/ (accessed 6.23.13).
[74] NSA, FBI secretly mines data from major Internet companies | The Daily Caller [WWW Document], n.d. URL http://dailycaller.com/2013/06/06/nsa-fbi-secretly-mines-data-from-major-Internet-companies/ (accessed 6.23.13).
[75] Office - Office.com [WWW Document], n.d. URL https://office.microsoft.com/en-us (accessed 6.23.13).
[76] Microsoft Visio 2013 – flowchart software - Office.com [WWW Document], n.d. URL https://office.microsoft.com/en-us/visio/business-and-software-diagrams-visio-professional-FX103472299.aspx (accessed 6.23.13).
[77] GitHub · Build software better, together. [WWW Document], n.d. URL https://github.com/ (accessed 6.23.13).
[78] Git [WWW Document], n.d. URL http://git-scm.com/ (accessed 6.23.13).
[79] The world's most popular free OS | Ubuntu [WWW Document], n.d. URL http://www.ubuntu.com/ (accessed 6.23.13).
[80] Tor Project: Anonymity Online [WWW Document], n.d. URL https://www.torproject.org/ (accessed 6.23.13).
[81] Pidgin, the universal chat client [WWW Document], n.d. URL http://pidgin.im/ (accessed 6.23.13).
[82] Python Programming Language – Official Website [WWW Document], n.d. URL http://www.python.org/ (accessed 6.23.13).
[83] Bitcoin - Open source P2P digital currency [WWW Document], n.d. URL http://bitcoin.org/en/ (accessed 6.23.13).
[84] Warlord territorial expansion [HARD VERSION] - Pastebin.com [WWW Document], n.d. URL http://pastebin.com/B2Gykp80 (accessed 6.23.13).

[85] Understanding Mining Bonds | Furuknap's Cryptocoin Blog [WWW Document], n.d. URL http://coin.furuknap.net/understanding-mining-bonds/ (accessed 6.23.13).
[86] CoinCleaner | Bitcoin laundry service [WWW Document], n.d. URL http://bxddobuy5lyotjqe.onion.to/ (accessed 6.23.13).
[87] Suport Bitcoin currency code BTC in ISO 4217 [WWW Document], n.d. URL https://bitcointalk.org/index.php?topic=7205.msg112577 (accessed 6.23.13).
[88] Trading Terms Glossary | RJO Futures [WWW Document], n.d. URL http://www.rjofutures.com/glossary/ (accessed 6.23.13).
[89] ^ Schulze, Hendrik; Klaus Mochalski (2009). "Internet Study 2008/2009". Leipzig, Germany: ipoque. Retrieved 2011-10-03. "Peer-to-peer file sharing (P2P) still generates by far the most traffic in all monitored regions – ranging from 43 percent in Northern Africa to 70 percent in Eastern Europe."

[90] Administración publica amortización de software
[91] Armory - Simple. Secure. Solid. [WWW Document], n.d. URL https://bitcoinarmory.com/ (accessed 6.23.13).